Irregular LDPC Codes Over GF(4) for CDMA Applications

A Thesis Submitted

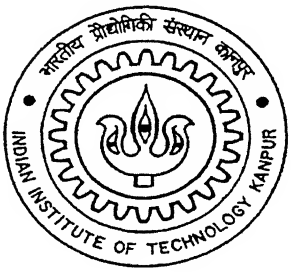
in Partial Fulfillment of the Requirements

for the Degree of

Master of Technology

by

Sachin Shrivastava



to the

DEPARTMENT OF ELECTRICAL ENGINEERING

INDIAN INSTITUTE OF TECHNOLOGY, KANPUR

February 2002

CERTIFICATE

It is certified that the work contained in the thesis entitled "*Irregular LDPC Codes Over GF(4) for CDMA Applications* " by *Sachin Shrivastava* has been carried out under my supervision and that this work has not been submitted elsewhere for a degree.

(A. K. Chaturvedi)

February 2002

Associate Professor,

Department of Electrical Engineering,

Indian Institute of Technology,

Kanpur-208016.

Acknowledgements

I consider myself to be privileged to have been supervised by Dr. A. K. Chaturvedi. His formidable insight and unfailing enthusiasm have proved a fertile source of stimulation and encouragement. He has always been generous with his time, listening carefully and criticising fairly.

This thesis is dedicated to my parents, my sister and my brother, whose importance to me I shall not try to put into words.

I would like to thank my friends specially Pankaj, Deshraj, Nagar, Ketan, Sharma Ji, Shashi, Bantu, Sampurnanand, Imran, Hari, Rajesh Ji and Sahu Ji for the warm affection and help they provided whenever I had some problem, and making my stay at IIT Kanpur a memorable one.

I would also like to thank Inference Group (http://www.inference.phy.cam.ac.uk) of David J. C. MacKay for providing help regarding LDPC codes. At various points I have had cause to be thankful to Vladislave Sorokine of Qualcomm Inc. and Matthew C. Davey to help me, whenever I had any doubt about LDPC codes.

I am very grateful to Igor Kozintsev of Intel Corp. for his assistance at various points in simulations, whenever I was in fix

Last but not the least, I would also like to thank God for showering upon me His blessings since my childhood

Sachin Shrivastava

Abstract

In this thesis we have analyzed the decoding complexity of sum-product algorithm for non-binary low-density parity-check (LDPC) codes. The FER performance of $GF(4)$ and $GF(2)$ codes in a binary Gaussian and fading channel has also been investigated. We have shown that using $GF(4)$ LDPC codes, we can get better results as compared to binary LDPC codes. Even with less decoding complexity $GF(4)$ codes outperform binary codes. We have also investigated the performance of $GF(4)$ LDPC codes in a CDMA cell environment.

Contents

1 Introduction 1

1.1 LDPC Codes 2

1.2 Spread Spectrum System 3

1.3 LDPC Codes for Spread-Spectrum Application 4

2 Low-Density Parity Check Codes 6

2.1 Introduction 6

2.2 Construction of Low-Density Parity Check Codes 7

2.2.1 Gallager's construction 7

2.2.2 MacKay's Constructions 8

2.2.3 Non-binary Codes 11

3 Decoding of LDPC Codes 13

3.1 Sum-product Algorithm 13

3.2 Transform Decoding 15

3.3 AWGN Channel 17

3.4 Rayleigh Fading Channel 19

3.4.1 CSI Available 19

3.4.2 CSI Not Available 20

4 Complexity Analysis and Decoder's Architecture **22**

4.1 Decoding Complexity 23

4.2 Decoder's Architecture 25

5 Results and Conclusions **31**

5.1 AWGN Channel 33

5.2 Rayleigh Fading Channel 38

5.2.1 CSI Available 39

5.2.2 CSI Not Available 41

5.3 Capacity of a CDMA Cell 45

5.4 Conclusions 47

5.5 Scope for Future Work 49

References **51**

List of Figures

1.1 Spread spectrum system using Rate 1/2 convolutional codes (a) Transmitter (b) Receiver. 3

2.1 Parity check matrix of Hamming Code 7

2.2 {20,3,4}Parity check matrix 8

2.3 Illustration of constructions of LDPC codes [10]. (a) construction 1A for code with $t_c = 3$, $t_r = 6$ and rate=1/2; (b) variant of Irregular construction for a code with rate 1/2; (c) for code of rate 1/4; (d) construction 2A for a code with rate 1/3. 9

2.4 Upper panels: fragments of equivalent parity-check matrices over (left) $GF(4)$ and (right) $GF(2)$. Lower panels: comparison of corresponding graph structure. Note the cycle in bold lines for binary code [10]. . . . 11

4.1 Decoder's Architecture : PU - Processor Unit, ICN - Interconnection networks, MM - Memory module. 27

4.2 Implementation of Vertical Step Processor Units $PU_i, i \in \{1, 2, 3, 4\}$, (a) for column of weight 2, (b) for column of weight 3. 28

4.3 Processor Unit PU_5 : CU - Control Unit. 29

4.4 Implementation of Hadamard Transform. 29

4.5 Implementation of Horizontal Step Computation (PU_6) for rows of (a) weight 3, $\{k1.k2\} = \{N(m) \setminus n\}$(b) weight 4, $\{k1.k2.k3\} = \{N(m) \setminus n\}$ 30

5.1 Weight distribution (fraction of codewords of given code weight) of rate rate 1/4 $GF(4)$ code N=384, K=96. 32

5.2 Frame error rates versus No. of Iterations for Rate 1/4 GF for Eb/No=1.5 dB in the AWGN channel. 34

5.3 Frame error rates for LDPC Codes in the AWGN channel. 35

5.4 Average number of iterations per frame in AWGN channel 36

5.5 Average number of iterations for correctly decoded frame in AWGN channel 37

5.6 Average number of operations required per information bit for AWGN channel. 39

5.7 Frame error rates versus No. of Iterations for Rate 1/4 GF(4) LDPC codes for Eb/No=3 dB in Rayleigh fading channel. The CSI is available to the decoder. 40

5.8 Frame error rates for LDPC Codes in the flat Rayleigh fading channel. The channel state information is made available to the decoder. 41

5.9 Average number of iterations per frame in Rayleigh fading channel The CSI is available to the decoder 42

5.10 Average number of iterations required for correctly decoded frame in Rayleigh fading channel. The CSI is available to the decoder. 43

5.11 Average number of operations required per information bit for Rayleigh fading channel. The CSI is available to the decoder. 44

5.12 Frame error rates for LDPC Codes in the flat Rayleigh fading channel with no CSI 45

5.13 Average number of iterations per frame in Rayleigh fading channel with no CSI 46

5.14 Average number of iterations required for correctly decoded frame in Rayleigh fading channel with no CSI 47

5.15 Average number of operations required per information bit for Rayleigh fading channel with no CSI 48

5.16 Data BER versus number of users in a cell for $GF(4)$ rate 1/4 code, binary rate 1/8 code [7], low-rate orthogonal convolutional code [12] . . 49

Chapter 1

Introduction

When data is transmitted over a noisy channel, errors occur. A major concern of a communication engineer is the control of these errors such that reliable transmission of data can be achieved. In 1948, Shannon demonstrated that by proper encoding and decoding, errors introduced by a noisy channel can be reduced to any desired level without sacrificing the data transmission rate, provided it is not greater than channel capacity.

In 1962, Gallager introduced Low-Density Parity-Check (LDPC) Codes, also known as Gallager Codes [1]. These codes are generally good only for low rates. At the time of their invention channel data rates were very moderate and hence LDPC codes were not considered useful.

However, after the introduction of spread spectrum based Cellular CDMA systems, LDPC codes had a rebirth. Viterbi [2] has shown that good performance can be achieved in CDMA system by allocating significant portion of the bandwidth expansion

to error control coding rather than PN or other sequences. Therefore CDMA system requires very low rate codes, but at very low rate decoding becomes impractical for ordinary block and convolutional codes. Good iterative decoding schemes exist for LDPC code at very low rate [3]. Hence LDPC codes can be used in spread-spectrum system to gain better performance.

1.1 LDPC Codes

Low-density parity-check codes are a class of linear error-correcting block codes. As the name suggests, parity check matrix of LDPC codes are very sparse, i.e. most of the elements of parity check matrix are zeros. They are defined in terms of a sparse parity check matrix: each codeword satisfies a large number of constraints, with each symbol of the codeword participating in a small number of constraints (e.g. 3).

In his original work, Gallager considered binary codes with same weight (e.g. 3) for each row and and same weight for each column. LDPC codes for which either all the rows of the parity check matrix do not have the same weight or/and all the columns do not have the same weight are known as irregular codes [4]. In this thesis we have considered irregular LDPC codes.

The main advantage of the LDPC codes are that these codes can be decoded using iterative decoders.

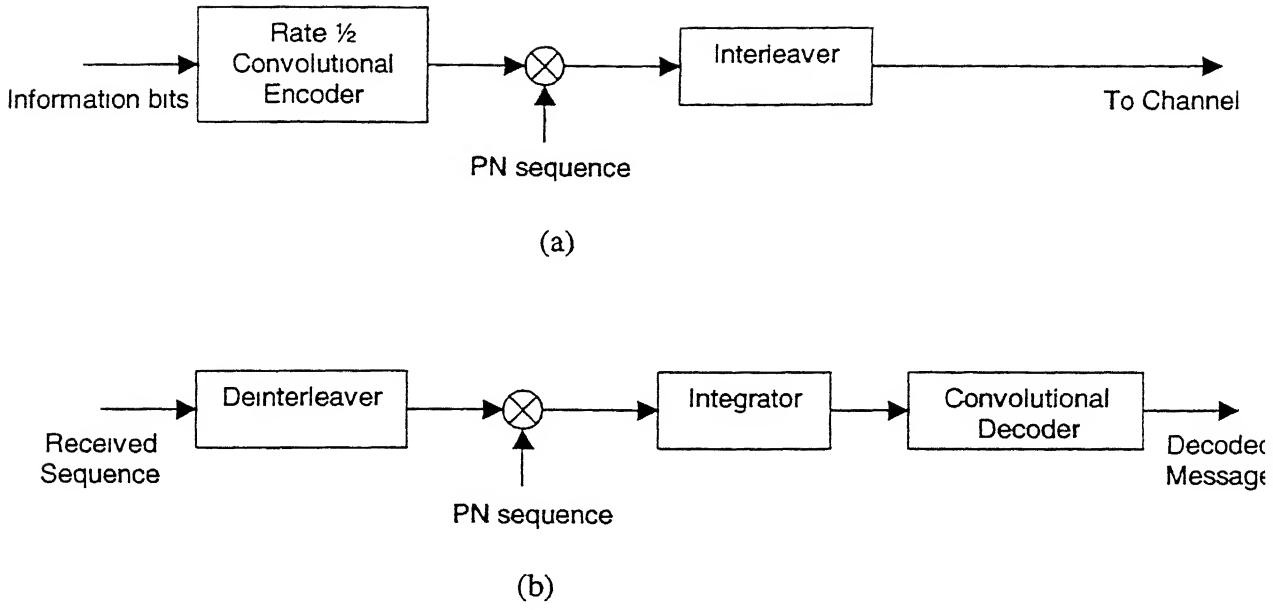


Figure 1.1: Spread spectrum system using Rate 1/2 convolutional codes (a) Transmitter (b) Receiver.

1.2 Spread Spectrum System

Spread-spectrum system have been used for achieving robustness against interference and jamming. As a multiple-access method, direct-sequence code-division multiple-access (DS-CDMA), where spreading is achieved by multiplying the signal with a pseudo random sequence, have received the maximum attention.

Fig. 1.1 shows the block diagram of the spread-spectrum system (e.g. IS-95 forward link) , using CDMA [5]. In the figure, rate 1/2 convolutional codes are used for error correction. Codewords are multiplied by PN sequence to combat multi-user interference and jamming. Interleaving is used to correct burst errors.

At receiver received signal is again multiplied by the same PN sequence and in-

tegrated. Interference from other users get canceled, since PN sequences for different users have little cross correlation.

In a conventional narrow-band communication system, any bandwidth expansion is primarily for the purpose of error correction. However, in spread-spectrum systems, higher capacity is achievable by employing low-rate channel codes for a large portion of the required bandwidth expansion. A limiting factor, has been the lack of good low rate error correcting codes. But LDPC codes perform very good at low rate, and can be decoded with low-complexity decoders.

1.3 LDPC Codes for Spread-Spectrum Application

Rate 1/2 convolutional codes as shown in Fig. 1.1 can be replaced by low rate LDPC codes. To make the overall spreading same, we need to reduce the spreading by PN sequence, so that the bit rate on the channel remains unchanged.

In [6] binary LDPC codes have been used for CDMA application. It is shown that binary LDPC codes outperform conventional CDMA systems, and CDMA systems using low-rate orthogonal convolutional codes in a multiaccess channel [7].

In this work instead of using binary codes as in [6], we have used irregular codes over $GF(4)$. It is shown that $GF(4)$ codes, can be decoded with very low-complexity decoders as compared to binary LDPC codes.

Following are the advantages of LDPC codes, that make them superior to convolutional and turbo codes.

- In terms of raw performance (bit error probability versus SNR), we can make LDPC codes that outperform Turbo codes [8] and convolutional codes [7].
- While Turbo and convolutional codes make undetected errors, LDPC codes do not make undetected errors [9].
- LDPC codes have lower decoding complexity.
- Fully-parallel hardware decoders can be made for LDPC codes.
- It is easy to create LDPC codes of almost any rate and blocklength. (It is hard to make arbitrary Turbo codes; you have to spend a lot of time fretting over the puncturing and searching for a good interleaver.)
- Unlike turbo codes LDPC codes are patent-free.

Therefore non-binary irregular LDPC codes could become the error-control scheme of choice in future generations of CDMA systems.

In chapter 2 we give a brief review of low-density parity-check codes. We describe several methods of constructing suitable low-density parity-check matrices.

In chapter 3 we discuss sum-product algorithm and transform decoding.

In chapter 4 we have done the complexity analysis of decoding of these codes. we also present the hardware architecture of decoder for these codes.

In chapter 5 we show simulation results of performance of these codes on AWGN and fading channels. We also show the performance in a multi-user environment. Chapter 5 also contains conclusions and scope for future work.

Chapter 2

Low-Density Parity Check Codes

2.1 Introduction

Low-density parity check codes are a class of linear block codes. In linear block codes a generator matrix G is used to generate a block t also known as a codeword corresponding to a message block s. All codewords satisfy $Ht = 0$, where H is the parity check matrix. Since these codes are linear codes, they also satisfy the property that sum of two codewords is also a codeword.

As their name suggests, parity-check matrix of low-density parity check codes is sparse, i.e. H consists almost entirely of zeros. According to Gallager's notation [1] $\{N, j, k\}$ is a linear code, where N is the length of codewords, and the matrix H has j number of ones in each column (the column property) and k number of ones in each row (the row property). If an H exists with either the row property only or the column property only, then we denote such codes as $\{N, -, k\}$ or $\{N, j, -\}$ codes, respectively, where the "blank" is used instead of the corresponding parameter. Note

$$\begin{bmatrix} 1 & 0 & 0 & 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 \end{bmatrix}$$

Figure 2.1: Parity check matrix of Hamming Code

that the parameter j and k in the notation $\{N, j, k\}$ characterize a particular parity-check matrix for code C. An equivalent parity-check matrix for the same code need not preserve the column or row property. The notation N, j, k rather describes the structure of a particular parity-check matrix for C.

For example Fig. 2.1 shows the Hamming code with parameter $(N = 7, K = 4)$ This H also qualifies as a parity-check matrix for a $\{7, -, 4\}$ code.

2.2 Construction of Low-Density Parity Check Codes

Many good codes can be constructed by specifying a fixed weight for each row and column, and constructing at random subject to constraints. However some specific constructions yield very good results. In this section we describe various constructions for low-density parity check codes for different rates, which have already been investigated in [1] and [9]

2.2.1 Gallager's construction

In his original work [1] Gallager imposed a fixed column weight j and a fixed row weight k The parity-check matrix was divided horizontally into j equal size submatrices, each

1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	1	1	1	1	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1
1	0	0	0	1	0	0	0	1	0	0	0	1	0	0	0	0	0	0	0
0	1	0	0	0	1	0	0	0	1	0	0	0	0	0	0	1	0	0	0
0	0	1	0	0	0	1	0	0	0	0	0	0	1	0	0	0	1	0	0
0	0	0	1	0	0	0	0	0	0	1	0	0	0	1	0	0	0	1	0
0	0	0	0	0	0	0	1	0	0	0	1	0	0	0	1	0	0	0	1
1	0	0	0	0	1	0	0	0	0	0	1	0	0	0	0	0	1	0	0
0	1	0	0	0	0	1	0	0	0	1	0	0	0	0	1	0	0	0	0
0	0	1	0	0	0	0	1	0	0	0	0	1	0	0	0	0	0	1	0
0	0	0	1	0	0	0	0	1	0	0	0	0	1	0	0	1	0	0	0
0	0	0	0	1	0	0	0	0	1	0	0	0	0	1	0	0	0	0	1

Figure 2.2: {20,3,4}Parity check matrix

containing a single '1' in each column. Without loss of generality the first submatrix was constructed in some predetermined manner (e.g. k concatenated identity matrices). The subsequent submatrices were random permutations of the first. A matrix constructed in this way is shown in Fig. 2.2.

2.2.2 MacKay's Constructions

A cycle in H is a sequence of distinct row-column indices $(r_1, c_1), (r_2, c_2), \ldots, (r_n, c_n), n$ even, with $r_1 = r_2, c_2 = c_3, r_3 = r_4$, etc., and $c_n = c_1$, and for each index (r_i, c_i), the

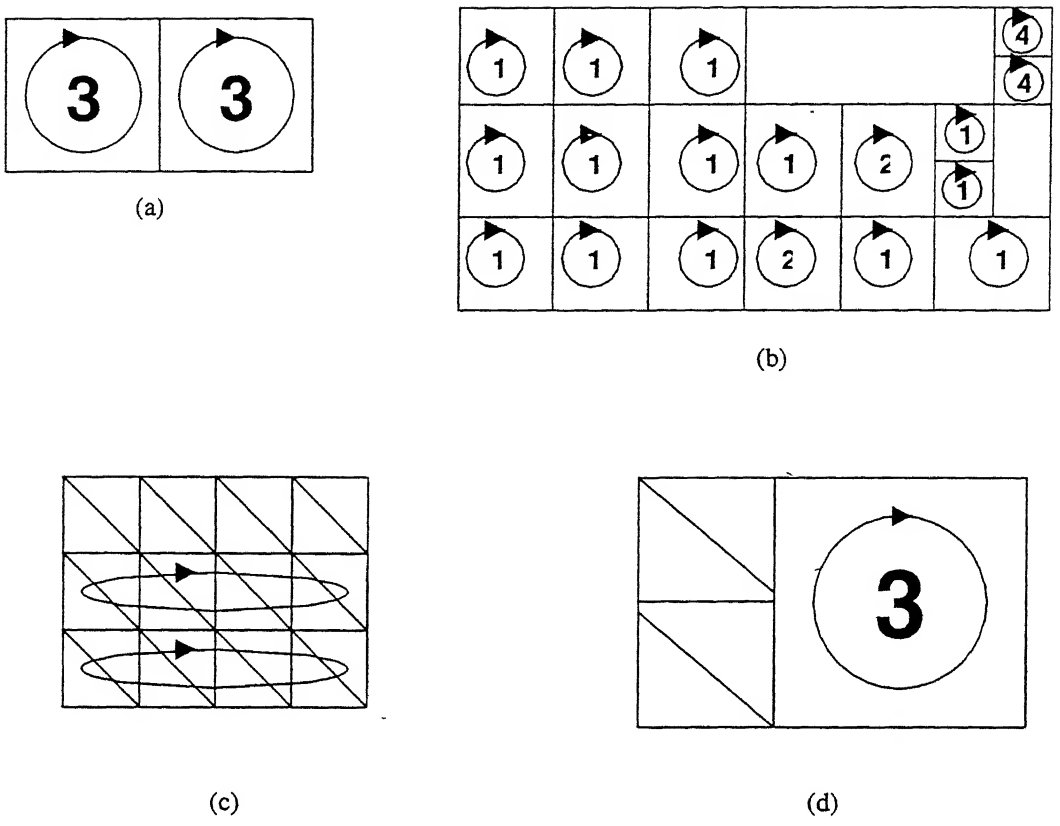


Figure 2.3: Illustration of constructions of LDPC codes [10]. (a) construction 1A for code with $t_c = 3$. $t_r = 6$ and rate=1/2; (b) variant of Irregular construction for a code with rate 1/2; (c) for code of rate 1/4; (d) construction 2A for a code with rate 1/3.

corresponding entry in H is nonzero. MacKay suggested some constructions to keep the number of short cycles as minimum as possible [9]. MacKay described following construction methods for matrices with no cycles of length 4. Let t_c represent the mean column weight and t_r represent the mean row weight of H. In case of matrix with *column property* $t_c = j$, and in case of *row property* $t_r = k$.

Notation In the Fig. 2.3 a diagonal represents an identity matrix A circle represents

a matrix constructed with column weight written inside circle, and keeping the row weight as uniform as possible. An ellipse imposed on a block within the matrix represents a random permutation of all the columns in that block.

Construction 1A This is a basic construction, in which we have a fixed weight per column t_c (e.g. 3) and construct the matrix at random keeping the weight per row (t_r) as uniform as possible, and overlap of non-zero terms between any two columns is not greater than 1. Shown in Fig. 2.3 (a)

Construction 2A As per 1A, except up to $M/2$,($M = N - K$, where K is the number of information symbols in the code) of the columns have weight 2. These weight 2 columns are constructed in the form of two identity matrices of size $M/2 \times M/2$, one above the other. Shown in Fig. 2.3 (d)

Construction 1B, 2B Some carefully chosen columns from 1A or 2A matrix are deleted, so that the bipartite graph of the matrix has no short cycle of length less than some length l, (e.g. , l=6).

MacKay has also suggested some irregular construction as shown in Fig. 2.3 (b). (Irregular codes, means that there exists, more than one row weights, or column weights).

Since parity check matrix as described earlier is not in systematic form, it can be converted to systematic form by performing Gaussian elimination using row operations and reordering of columns to derive a parity check matrix $H' = [P|I_m]$ where the notation $[A|B]$ indicates the concatenation of matrix A with B, and I_m represents the $M \times M$ identity matrix. We now redefine the original H to include the column

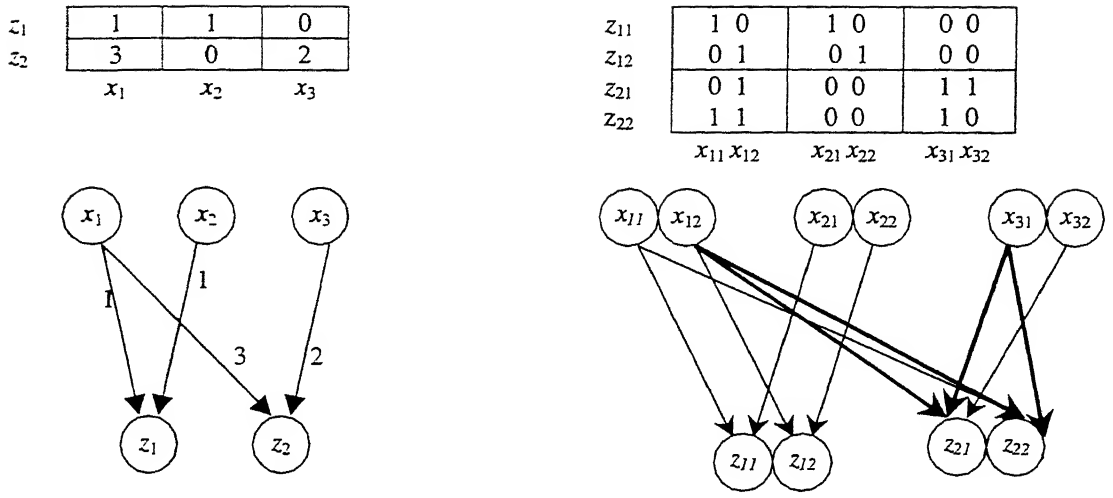


Figure 2.4: Upper panels: fragments of equivalent parity-check matrices over (left) $GF(4)$ and (right) $GF(2)$. Lower panels: comparison of corresponding graph structure. Note the cycle in bold lines for binary code [10].

reordering as per the Gaussian elimination. The corresponding generator matrix is then $G = [I_k|P]$.

2.2.3 Non-binary Codes

Non-binary codes means elements of the H matrix are taken from $GF(q)$. $GF(q)$ codes can be constructed from binary codes, by replacing all ones, by elements of $GF(q)$ according to a carefully selected random distribution; rather than using the uniform distribution We choose the entries in each row to maximize the entropy of the corresponding symbol of the syndrome vector $z = Hx$ where x is a sample from the assumed channel noise model. We refer to elements of x as noise symbols and elements of z as checks

Non-binary LDPC codes are known to perform better. In LDPC codes undetected errors do not occur i.e error occurs whenever decoding does not converge The main reason of non-convergence are the short cycles in the bipartite graph generated according to H matrix Therefore H matrix should not have short cycles but as the weight increases, it is not possible to construct the matrix without cycles in $GF(2)$

But in $GF(q)$ for $q > 2$ it is possible to construct H matrix without short cycles [10] Fig. 2.4 shows the graphs for two equivalent parity-check matrix fragments We can see that the q-ary code contains no cycles, whereas the binary code has a cycle of length 4 (bold lines). Thus we can increase the density of the H matrix without increasing the number of short cycle in $GF(q)$ Increase in density means more number of checks, which improves the performance of the code

Chapter 3

Decoding of LDPC Codes

Having encoded the source vector s using the generator matrix G we transmit the message $t = G^T s$. The channel introduces noise and we receive the vector $r = t + n$, where n is the noise vector. We multiply our received vector by H to form the syndrome vector $z = Hr = HG^T s + Hn = Hn$, where all the arithmetic is performed over a finite field, depending on the codes used. We perform syndrome decoding. That is, we want to find the most probable vector x (according to the channel model) that explains the observed syndrome vector $z = Hx$. The estimate of the noise vector is then x.

3.1 Sum-product Algorithm

Sum-product algorithm is the most popular algorithm used for decoding of LDPC codes. This algorithm is a generalization of the approximate belief propagation algorithm used by Gallager [1].

We will refer to elements of x as noise symbols and elements of z as checks. Let

$N(m) = \{n : H_{mn} \neq 0\}$ be the set of indices of noise symbols that participate in check m Let $M(n) = \{m : H_{mn} \neq 0\}$ be the set of indices of checks that depend on noise symbol n.

With each nonzero entry in the parity check matrix H_{mn} we associate q^a_{mn}, r^a_{mn} for each $a \in GF(q)$. The quantity q^a_{mn} is the probability that symbol n of x is a, given the information obtained via checks other than check m, because we want the information to propagate instead of coming back to the same symbol This is done, because if it comes back to the same symbol, it will mean a cycle of length 2. The quantity r^a_{mn} is the probability of check m being satisfied if symbol n of x is considered fixed at a and the other noise symbols have a separable distribution given by the probabilities $\{q^a_{mn'} : n' \in N(m) \setminus n, a \in GF(q)\}$.

A Initialization

We initialize the values of q^a_{mn} to f^a_n, the likelihood that $x_n = a$ according to the channel model The expressions for finding channel likelihoods are derived in section 3 3 and 3.4 for AWGN and Rayleigh fading channel respectively.

B Updating r^a_{mn} (Horizontal Step)

At each iteration the value of r^a_{mn} is given by

$$r^a_{mn} = \sum_{x' \; x'_n = a} Pr[z_m|x'] \prod_{j \in N(m) \setminus n} q^{x'_j}_{mj} \tag{3.1}$$

where $Pr[z_m|x'] \in \{0, 1\}$ according to whether or not x' satisfies check m, i.e. if m'th symbol of syndrome generated by x' is same as z_m then $Pr[z_m|x'] = 1$, else

$Pr[z_m|x'] = 0$.

C. Updating q^a_{mn} (Vertical Step)

For each m, n and $a \in GF(q)$ we compute

$$q^a_{mn} = \alpha_{mn} f^a_n \prod_{j \in M(n) \backslash m} r^a_{jn} \tag{3.2}$$

where α_{ij} a normalizing factor is chosen such that $\sum_a q^a_{ij} = 1$. The normalizing factor α_{mn} is convenient but not essential for decoding.

We then make a tentative decoding $\hat{x}$ such that

$$\hat{x} = \overset{argmax}{a} f^a_n \prod_{j \in M(n)} r^a_{jn} \tag{3.3}$$

which means choosing the value of $\hat{x}_n$ to be that a for which $f^a_n \prod_{j \in M(n)} r^a_{jn}$ is maximum.

If $H\hat{x} = z$ then the decoding algorithm halts having identified a valid decoding of the syndrome, otherwise the algorithm repeats. A failure is declared if some maximum number of iterations occurs without a valid decoding.

Although it is in principle possible for the decoder to converge to the wrong noise vector, but it has not been observed in our simulations. Thus empirically all decoding errors are detected, which is in agreement with [10].

3.2 Transform Decoding

The complexity of the horizontal step of sum-product algorithm scales as q^2 for non-binary LDPC codes over $GF(q)$. This complexity can be reduced using a Hadamard transform of the probabilities, as suggested in [10].

From (3.1) we can write the horizontal step as:

$$r^a_{mn} = \sum_{x':x'_n=a} Pr\left(\sum_{n'\in N(m)} H_{mn'}x'_{n'} = z_m\right) \prod_{j\in N(m)\setminus n} q^{x'_j}_{mj} \tag{3.4}$$

This represents a convolution of the quantities q^a_{mj}, and so the summation can be replaced by a product of the Hadamard transforms (taken over the additive group of $GF(q)$) of q^a_{mj} for $j \in N(m) \setminus n$, followed by an inverse Hadamard transform. The Hadamard transform F of a function f over $GF(2)$ is given by $F^0 = f^0 + f^1$, $F^1 = f^0 - f^1$. Transforms over $GF(2^p)$ can be viewed as a sequence of binary transforms in each of p dimensions. Hence for $GF(2^2)$ we have

$$F^0 = [f^0 + f^1] + [f^2 + f^3] \tag{3.5}$$

$$F^1 = [f^0 - f^1] + [f^2 - f^3] \tag{3.6}$$

$$F^2 = [f^0 + f^1] - [f^2 + f^3] \tag{3.7}$$

$$F^3 = [f^0 - f^1] - [f^2 - f^3] \tag{3.8}$$

The inverse transform is the same, except that it is subsequently divided by 2^p.

Let vector $\mathbf{Q}_{mj} = (Q^0_{mj}, Q^1_{mj}, \ldots, Q^{q-1}_{mj})$ represent the Hadamard transform of the vector $\mathbf{q}_{mj} = (q^0_{mj}, q^1_{mj}, \ldots, q^{q-1}_{mj})$. Then the inverse transform of

$$\left(\left(\prod_{j\in N(m)\setminus n} Q^0_{mj}\right), \ldots, \left(\prod_{j\in N(m)\setminus n} Q^{q-1}_{mj}\right)\right) \tag{3.9}$$

gives $(r^0_{mn}, \ldots, r^{q-1}_{mn})$.

3.3 AWGN Channel

Let us find the channel likelihood for the AWGN channel. The received signal is

$$y = s + x \tag{3.10}$$

where s is the transmitted signal (± 1), x is the white Gaussian noise with variance σ^2. We define the received bit to be the sign of the channel output.

Now a posterior probability that 1 was transmitted given y is

$$\begin{aligned} Pr(s=1|y) &= \frac{Pr(y|s=1)Pr(s=1)}{Pr(y)} \\ &= \frac{Pr(y|s=1)Pr(s=1)}{Pr(s=1)Pr(y|s=1) + Pr(s=-1)Pr(y|s=-1)} \end{aligned} \tag{3.11}$$

We assume that $Pr(s=1) = Pr(s=-1) = 1/2$. Hence

$$Pr(s=1|y) = \frac{1}{1 + \frac{Pr(y|s=-1)}{Pr(y|s=1)}} \tag{3.12}$$

Let us assume that -1 (symbol 0) is transmitted, then distribution of y will be Gaussian with mean -1 and variance σ^2. Hence,

$$Pr(y|s=-1) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(\frac{-(y+1)^2}{2\sigma^2}\right) \tag{3.13}$$

Similarly

$$Pr(y|s=1) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(\frac{-(y-1)^2}{2\sigma^2}\right) \tag{3.14}$$

Putting (3.13) and (3.14) in (3.12), we get

$$Pr(s=1|y) = \frac{1}{1+\frac{\exp\left(\frac{-(y+1)^2}{2\sigma^2}\right)}{\exp\left(\frac{-(y-1)^2}{2\sigma^2}\right)}} \tag{3.15}$$

$$= \frac{1}{1+\exp(-2y/\sigma^2)} \tag{3.16}$$

Similarly

$$Pr(s=-1|y) = \frac{1}{1+\exp(2y/\sigma^2)} \tag{3.17}$$

We denote channel likelihoods by g^1 and g^0 where g^1 is the a posterior probability that noise bit is 1 and g^0 is the a posterior probability that noise bit is 0. Now if received y is positive, channel likelihoods can be expressed as

$$g^1 = Pr(s=-1|y) \tag{3.18}$$

$$g^0 = Pr(s=1|y) \tag{3.19}$$

and if received y is negative

$$g^1 = Pr(s=1|y) \tag{3.20}$$

$$g^0 = Pr(s=-1|y) \tag{3.21}$$

Hence for n'th symbol, combining (3.16-3.21)

$$g_n^1 = \frac{1}{1+\exp(2|y_n|/\sigma^2)} \tag{3.22}$$

where y_n is the output of the channel [8]. Now g_n^0 can be found using g_n^0=1-g_n^1.

In $GF(2^b)$ each noise symbol x_n consists of b noise bits $x_{n_1}, \ldots, x_{n_b}$. We consider a memoryless binary channel so we can set the likelihood of the noise symbol x_n being

equal to a to $f_n^a = \prod_{i=1}^{b} g_{n_i}^{a_i}$ for each $a \in GF(q)$ where a_i is the i'th bit of the binary representation of a.

3.4 Rayleigh Fading Channel

In this section we find the channel likelihoods for the case of Rayleigh fading channel. We consider flat Rayleigh-fading channel with independent fading amplitudes and unity second moment. Although this assumption is often not valid in practical situations, but it can be achieved using interleaving.

We considered ± 1 antipodal signalling. We assumed perfect synchronization at the receiver. Then the received signal is

$$y = \alpha s + x \tag{3.23}$$

where s is the transmitted signal (± 1), x is the white Gaussian noise with variance σ^2 and α is a Rayleigh random variable such that $E[\alpha^2]$ is 1.

For fading channel we have considered two different cases, one when α i.e. fading amplitude is known to receiver, and other when α is unknown to the receiver.

3.4.1 CSI Available

In this case we assume that the channel state information (CSI) is available to the decoder, i.e. the decoder knows the fading amplitudes α for each transmitted symbol.

Channel likelihood for this case can be found as for the AWGN case in (3.22).

$$g_n^1 = \frac{1}{1+\exp(2\alpha|y_n|/\sigma^2)} \tag{3.24}$$

3.4.2 CSI Not Available

This is a more realistic situation, since generally receiver doen not have knowledge of fading amplitude. Instead we assume that the receiver knows the second moment of fading (i.e. $E[\alpha^2]$).

As in (3.23)

$$y = \alpha s + x \tag{3.25}$$

Let us assume that -1(symbol 0) is transmitted, then distribution of y will be Gaussian with mean $-\alpha$, where α is Rayleigh distributed with $E[\alpha^2] = 1$. Hence,

$$Pr(y|\alpha, s=-1) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(\frac{-(y+\alpha)^2}{2\sigma^2}\right) \tag{3.26}$$

$$Pr(y|s=-1) = \int_0^\infty f(\alpha) Pr(y|\alpha, s=-1) d\alpha \tag{3.27}$$

$$Pr(y|s=-1) = \int_0^\infty f(\alpha) \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(\frac{-(y+\alpha)^2}{2\sigma^2}\right) d\alpha \tag{3.28}$$

where $f(\alpha)$ is Rayleigh distribution, such that $E[\alpha^2] = 1$,

$$f(\alpha) = 2\alpha \exp(-\alpha^2) \qquad ; \alpha = [0, \infty] \tag{3.29}$$

Hence,

$$Pr(y|s=-1) = \int_0^\infty 2\alpha \exp(-\alpha^2) \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(\frac{-(y+\alpha)^2}{2\sigma^2}\right) d\alpha \tag{3.30}$$

Similarly,

$$Pr(y|s=1) = \int_0^\infty 2\alpha \exp(-\alpha^2) \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(\frac{-(y-\alpha)^2}{2\sigma^2}\right) d\alpha \tag{3.31}$$

Putting (3.30) and (3.31) in (3.12), we get

$$Pr(s=1|y) = \frac{1}{1 + \frac{\int_0^\infty 2\alpha \exp(-\alpha^2) \exp\left(\frac{-(y+\alpha)^2}{2\sigma^2}\right) d\alpha}{\int_0^\infty 2\alpha \exp(-\alpha^2) \exp\left(\frac{-(y-\alpha)^2}{2\sigma^2}\right) d\alpha}} \tag{3.32}$$

Similarly

$$Pr(s=-1|y) = \frac{1}{1 + \frac{\int_0^\infty 2\alpha \exp(-\alpha^2) \exp\left(\frac{-(y-\alpha)^2}{2\sigma^2}\right) d\alpha}{\int_0^\infty 2\alpha \exp(-\alpha^2) \exp\left(\frac{-(y+\alpha)^2}{2\sigma^2}\right) d\alpha}} \tag{3.33}$$

Now if received y is positive, channel likelihoods can be found as

$$g^1 = Pr(s=-1|y) \tag{3.34}$$

$$g^0 = Pr(s=1|y) \tag{3.35}$$

and if received y is negative

$$g^1 = Pr(s=1|y) \tag{3.36}$$

$$g^0 = Pr(s=-1|y) \tag{3.37}$$

Combining (3.32-3.37), we can express the result as

$$g^1 = \frac{1}{1 + \frac{\int_0^\infty \alpha \exp(-\alpha^2) \exp\left(\frac{-(|y|-\alpha)^2}{2\sigma^2}\right) d\alpha}{\int_0^\infty \alpha \exp(-\alpha^2) \exp\left(\frac{-(|y|+\alpha)^2}{2\sigma^2}\right) d\alpha}} \tag{3.38}$$

and $g^0 = 1 - g^1$.

Chapter 4

Complexity Analysis and Decoder's Architecture

In this chapter we have found the analytical expression of complexity involved in sum-product decoding algorithm of non-binary irregular LDPC codes.

Later in the chapter we discuss one of the most important issue that determines the feasibility of a practical coding scheme. Whereas there are many good block and convolutional codes whose performance is known to be good, a restricting factor is the unavailability of low-cost hardware that is required to extract this performance. We provide hardware blocks that could be used to implement the sum-product algorithm decoder. Hardware implementation discussed in this section builds upon the hardware implementation of sum-product algorithm for binary codes discussed in [7]. We have modified the implementation by including transform decoding and generalizing it from binary to $GF(4)$.

4.1 Decoding Complexity

In this section we discuss the computational complexity of the sum product algorithm involving the transform domain approach explained in section 3.2. An important merit of LDPC codes compared to other block and convolutional codes is the existence of low-complexity decoders.

The computational complexity of the sum-product algorithm is directly proportional to the number of nonzero elements in the H matrix. For non-binary codes in $GF(q)$ complexity varies as q^2. Therefore for a code in $GF(q)$ of block length N, mean column weight t_c, computational complexity varies as Nt_cq^2. In our analysis we have considered irregular codes over $GF(q)$.

Let u_t represent the number of rows of weight t and v_t represent the number of columns of weight t in H.

For calculating the computational complexity of horizontal steps using transform decoding as described in section 3.2, we need to take into account all parity checks. For each parity check number of operations (our definition of operation is the same as used in [7] i.e. each multiplication or addition is considered one operation) required to calculate transform of $\mathbf{q}_{mn}$ are $q\log(q)$ as can be seen from (3.5-3.8) for $GF(4)$. Total number of operations required to implement (3.9), are $q(t-2)$ for rows of weight t. The number of operations required to calculate the inverse transform of (3.9) is $q\log(q)+q$, per check. So number of operations required in horizontal steps are

$$C_H = \sum_{t=1}^{N} tu_t[2q\log(q) + q + q(t-2)] \tag{4.1}$$

Let $M(n) = \{m : H_{mn} \neq 0\} = \{k_1, k_2, \ldots, k_t\}$ where t is the weight of the n'th column. As defined earlier f_n^a is the probability of n'th noise bit being a. Now vertical steps (3.2) can be implemented as

$$q_{k_1 n}^a = f_n^a \times r_{k_t n}^a \times r_{k_{t-1} n}^a \times \ldots\ldots\ldots\ldots\ldots r_{k_2 n}^a \tag{4.2}$$

$$q_{k_2 n}^a = f_n^a \times r_{k_t n}^a \times r_{k_{t-1} n}^a \times \ldots\ldots\ldots\ldots r_{k_3 n}^a \times r_{k_1 n}^a \tag{4.3}$$

$$q_{k_3 n}^a = f_n^a \times r_{k_t n}^a \times r_{k_{t-1} n}^a \times \ldots r_{k_4 n}^a \times r_{k_2 1 n}^a \times r_{k_1 1 n}^a \tag{4.4}$$

$\ldots$

$$q_{k_t n}^a = f_n^a \times r_{k_{t-1} n}^a \times r_{k_{t-2} n}^a \times \ldots r_{k_3 n}^a \times r_{k_2 1 n}^a \times r_{k_1 1 n}^a \tag{4.5}$$

In the above equations we have not used α_{mn} of (3.2) since it is not essential. Computation of $q_{k_1 n}^a$ in (4.2) requires $(t-1)$ multiplications. Now $q_{k_2 n}^a$ in (4.3) can be calculated by multiplying $f_n^a \times r_{k_t n}^a \times r_{k_{t-1} n}^a \times \ldots \times r_{k_3 n}^a$ (this is available from (4.2)) and $r_{k_1 n}^a$ with just one operation. Similarly $q_{k_3 n}^a$ in (4.4) can be calculated using two operations by multiplying $f_a^n \times r_{k_t n}^a \times r_{k_{t-1} n}^a \ldots \times r_{k_4 n}^a$ (available from (4.2)) with $r_{k_2 n}^a$ and $r_{k_1 n}^a$. Similarly other $r_{k_p n}^a$,$(3 \leq p \leq t)$ can be calculated by multiplying $f_n^a \times r_{k_t n}^a \times r_{k_{t-1} n}^a \times \ldots r_{k_{p+1} n}^a$ (available from (4.2)) , $r_{k_{p-1} n}^a$ and $r_{k_{p-2} n}^a \times \ldots \times r_{k_2 n}^a \times r_{k_1 n}^a$ (available from computation of $q_{k_{p-1} n}^a$) with only two operations needed. So total operations required by a column of weight t are $(t-1) + 1 + 2(t-2) = 3t - 4$.

Hence, total number of operations required for vertical steps are

$$C_V = q \sum_{i=1}^{N-K} v_t(3t-4) \tag{4.6}$$

Computation of the tentative posterior probabilities in (3.3) will require only qN operations as these can be obtained by multiplying (3.2) by appropriate column element $r_{k_l}^a$.

Thus the total complexity of the sum-product algorithm per iteration, per information bit is

$$C = \frac{C_H + C_V + qN}{K} \tag{4.7}$$

where K is the length of information sequence.

4.2 Decoder's Architecture

In this section we illustrate the hardware implementation of the decoder for $GF(4)$ codes with row weight of 3 or 4, and column weight of 2 or 3.

The decoder consists of six processor units and three interconnection networks (ICN) that form a pipeline design. Various ICNs shown in the figures are required to rearrange the signal in the desired form required by the next processor. These ICNs can be formed by using Banyan Networks [7]. These networks are programmed only once for each LDPC code. A memory module is used as data buffer. The schematic configuration that illustrates parallelism and pipelineability of the decoder is shown in

Fig 4 1 $\mathbf{Q^a}$ and $\mathbf{R^a}$ are the matrix with elements q_{ij}^a and r_{ij}^a respectively, where q_{ij}^a and r_{ij}^a are as defined in chapter 3

The signal flow graph for the processor unit PU_i, $i \in 1, 2, 3, 4$ is shown in Fig 4 2 This processor unit executes the vertical step (3 2) of the sum product algorithm Since the number of nonzero elements in a column of H is either 2 or 3, the processor unit consists of two parts, top part (a) for column of weight 2 and bottom part (b) for column of weight 3. In the figure f_i are the likelihood functions for i th received codeword

Fig. 4.3 shows the processor unit PU_5 It performs the syndrome computation of the tentative noise vector n and the received vector y and compares these syndromes. If they are identical, the control unit CU gives a command to output the decoded vector x and load new f^0, f^1, f^2, f^3 (i.e. start decoding of a new vector y). If the syndromes are not identical, PU_5 outputs the $\mathbf{Q^0}, \mathbf{Q^1}, \mathbf{Q^2}, \mathbf{Q^3}$ which are used in the next iteration for the same y.

The signal flow graph for the processor PU_6 is shown in Fig. 4.5. This processor unit computes the horizontal step. Since the number of nonzero elements in a row of H is either 3 or 4, the processor unit consists of two parts, top part (a) for rows of weight 3 and bottom part (b) for rows of weight 4. Input to the Fig. 4 5 is the Hadamard transform of the q_{ij}^as which can be computed as shown in Fig. 4.4. The two blocks shown in the Fig. 4.5 are for computing Hadamard Transform as shown in Fig. 4.4 according to (3.5-3 8)

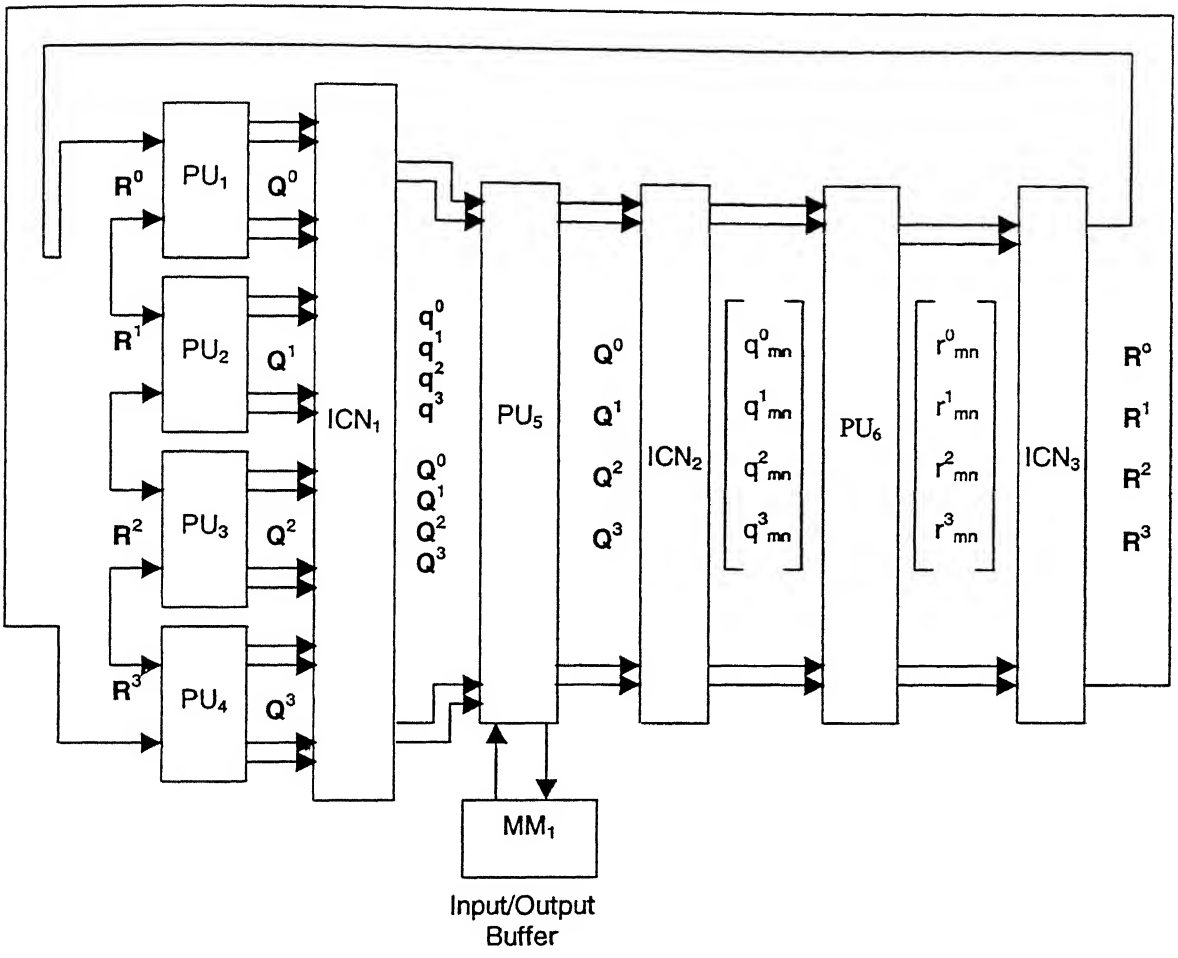
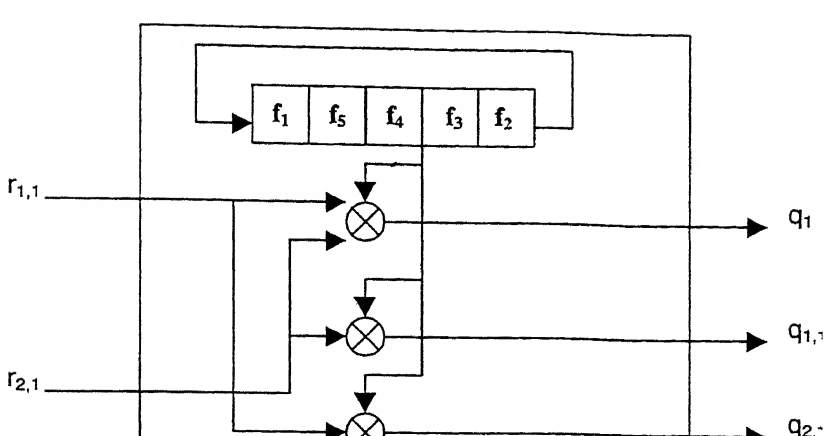
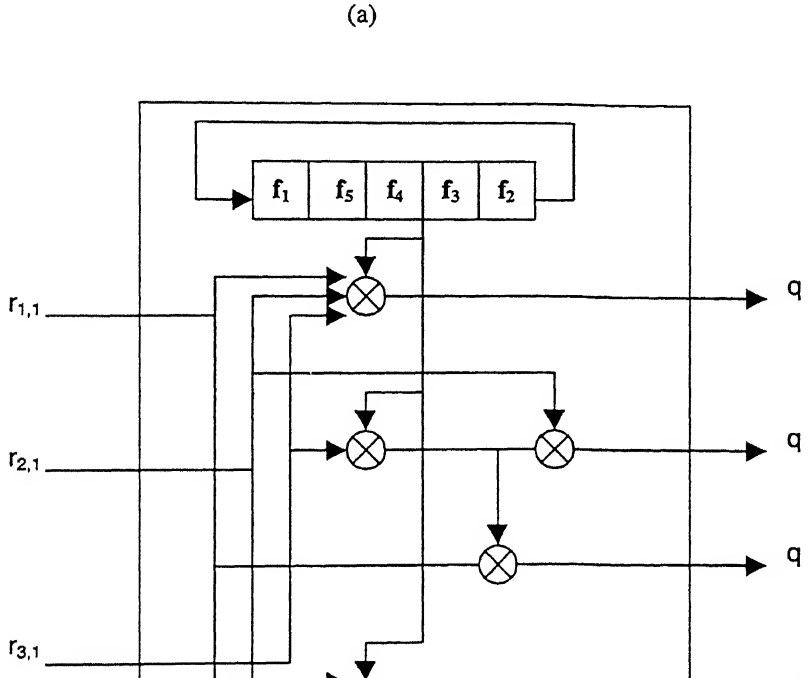


Figure 4.1: Decoder's Architecture : PU - Processor Unit, ICN - Interconnection networks, MM - Memory module.



(a)



(b)

Figure 4.2: Implementation of Vertical Step Processor Units $PU_i, i \in \{1, 2, 3, 4\}$. (a) for column of weight 2. (b) for column of weight 3.

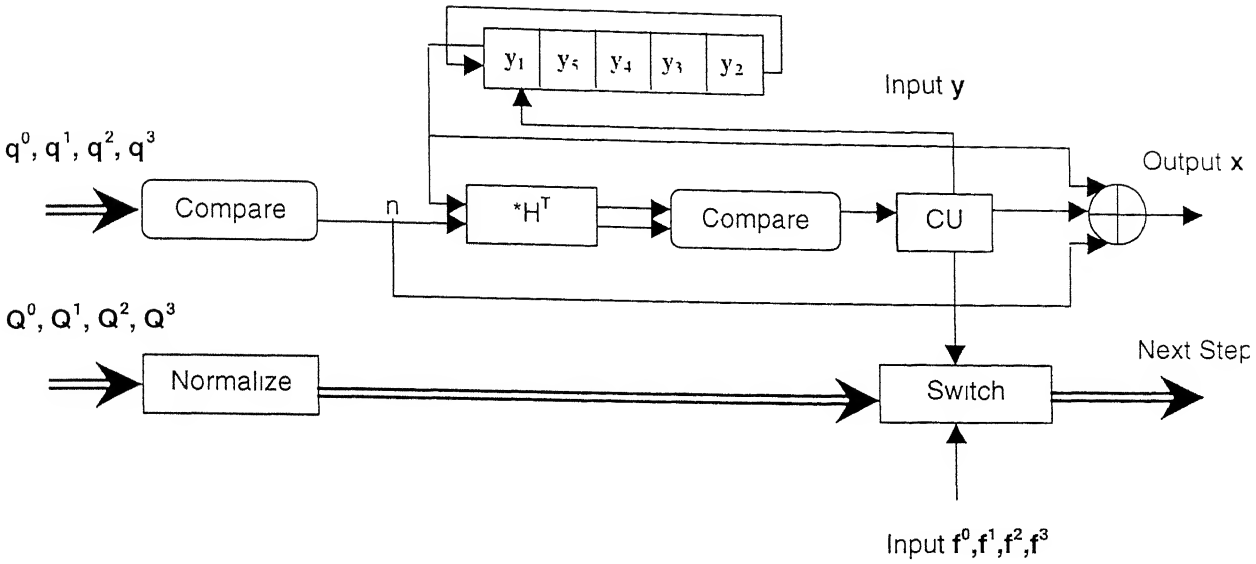


Figure 4.3: Processor Unit PU; CU - Control Unit

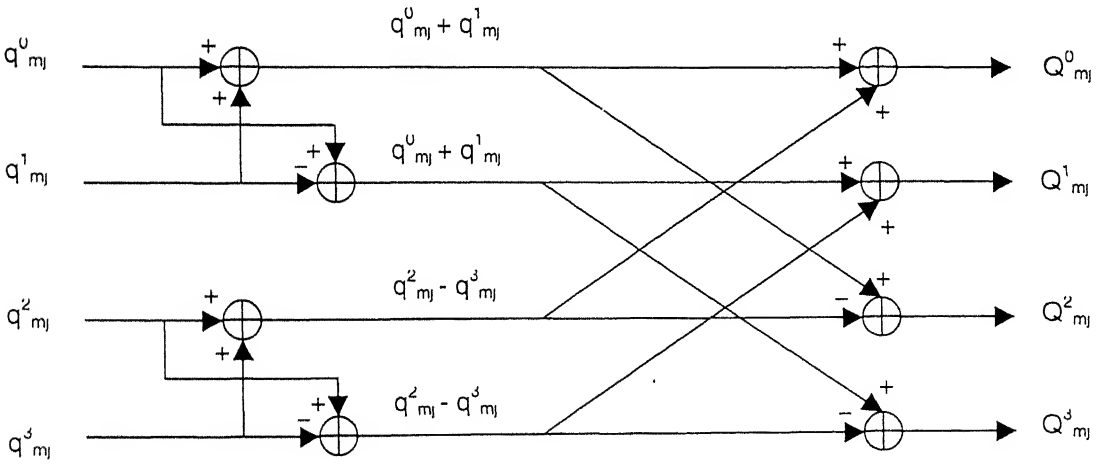


Figure 4.4: Implementation of Hadamard Transform

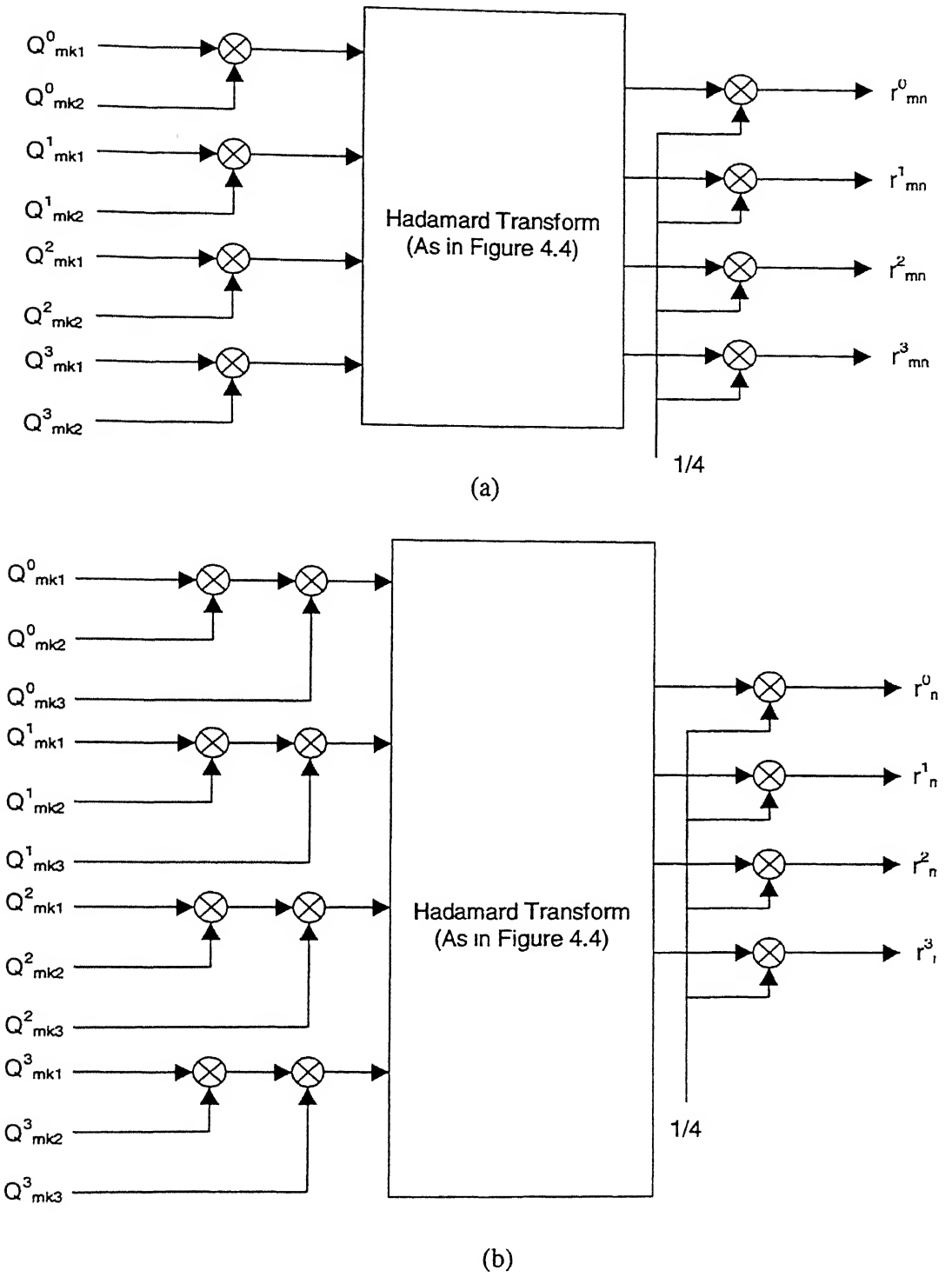


Figure 4.5. Implementation of Horizontal Step Computation (PU_6) for rows of (a) weight 3. $\{k1, k2\} = \{N(m) \setminus n\}$(b) weight 4. $\{k1, k2, k3\} = \{N(m) \setminus n\}$

Chapter 5

Results and Conclusions

In this chapter we have shown the simulation results for a rate 1/4 $GF(4)$ irregular LDPC code, in AWGN and Rayleigh fading channel. We have used (384,96) code with the parity check matrix of size 288×384. Thus the number of information symbols per frame is 96 which corresponds to 192 information bits. This choice makes the code compatible with the IS-95 cellular CDMA system frame size of 192 bits [5]. To construct H we have used the method as described in [9]. Parameters of the parity check matrix used are as following

Mean row weight = 3.5

Mean Column Weight = 2.625

No. of rows of weight 3 = u_3 = 144

No. of rows of weight 4 = u_4 =144

No. of columns of weight 2 = v_2 = 144

No. of columns of weight 3 = v_3 = 240

Density of H matrix = 0.0091

Weight distribution of a code can be used to find the fraction of codewords of a given weight. Fig. 5.1 shows the weight distribution of the code that we have used. We have used Monte-Carlo simulation to estimate the weight distribution. We generated large number of random message sequences over $GF(4)$ of length 96 and encoded using generator matrix to obtain the codewords of length 384. From these ensembles of codewords we found the fraction of codewords of a given weight.

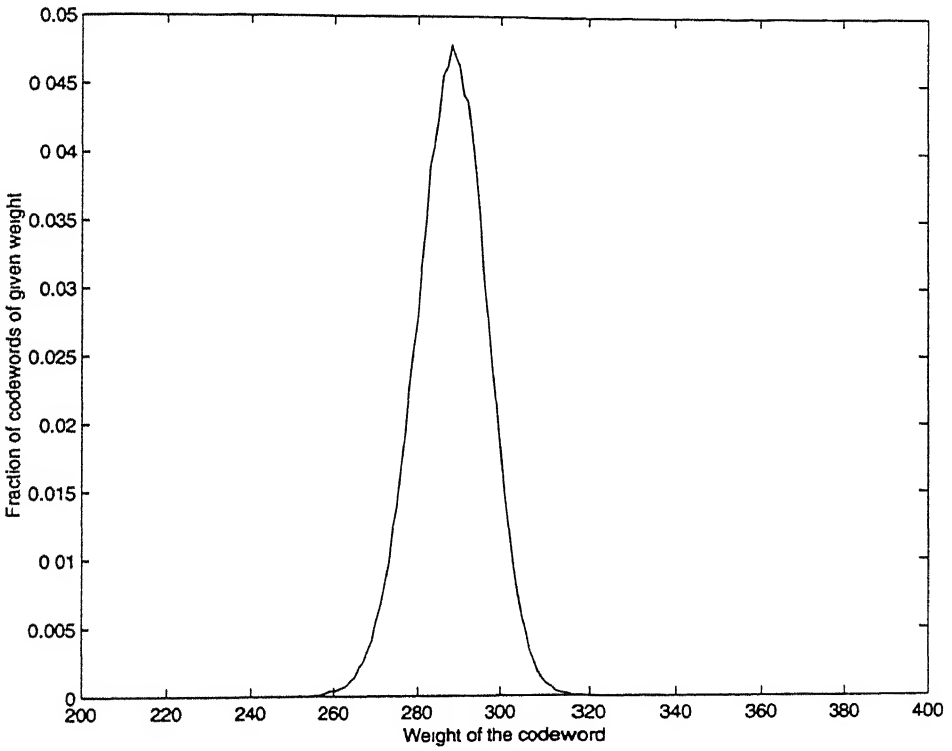


Figure 5.1: Weight distribution (fraction of codewords of given code weight) of rate rate 1/4 $GF(4)$ code N=384, K=96.

For comparison with binary codes, we have used the rate 1/8 binary (1536,192)

code of [6].

The reason for choosing the binary code to be rate 1/8 was twofold. First if we choose a higher rate code the performance is going to be inferior. Rate 1/8 codes have been shown to provide good performance in [7]. Secondly we did not want to choose a lower rate code because that would make the complexity very high, even otherwise simulation shows that the performance of rate 1/16 codes was inferior to rate 1/8 codes.

By putting code parameters in (4.7), per iteration per information bit complexity is found to be 177 for the rate 1/4 $GF(4)$ codes, while complexity per iteration per information bit for rate 1/8 binary code is 162 [7].

5.1 AWGN Channel

We have applied our codes to a binary Gaussian channel with inputs ± 1 and additive noise of variance σ^2. Thus on the channel each $GF(4)$ symbol is transmitted as two binary symbols, making no use of algebraic structure of $GF(4)$. If the code rate is R then E_b/N_o is given by $1/2R\sigma^2$ [8]. The received signal is given by

$$y = s + x \tag{5.1}$$

where s is the transmitted signal (± 1), x is the white Gaussian noise with variance σ^2. We define the received bit to be the sign of the channel output. We have used the channel likelihoods as derived in section 3.3.

Fig. 5.2 shows the effect of number of iterations on Frame error rate (FER) for E_b/N_o=1.5 dB. It is clear from the figure that we can not gain any significant advantage

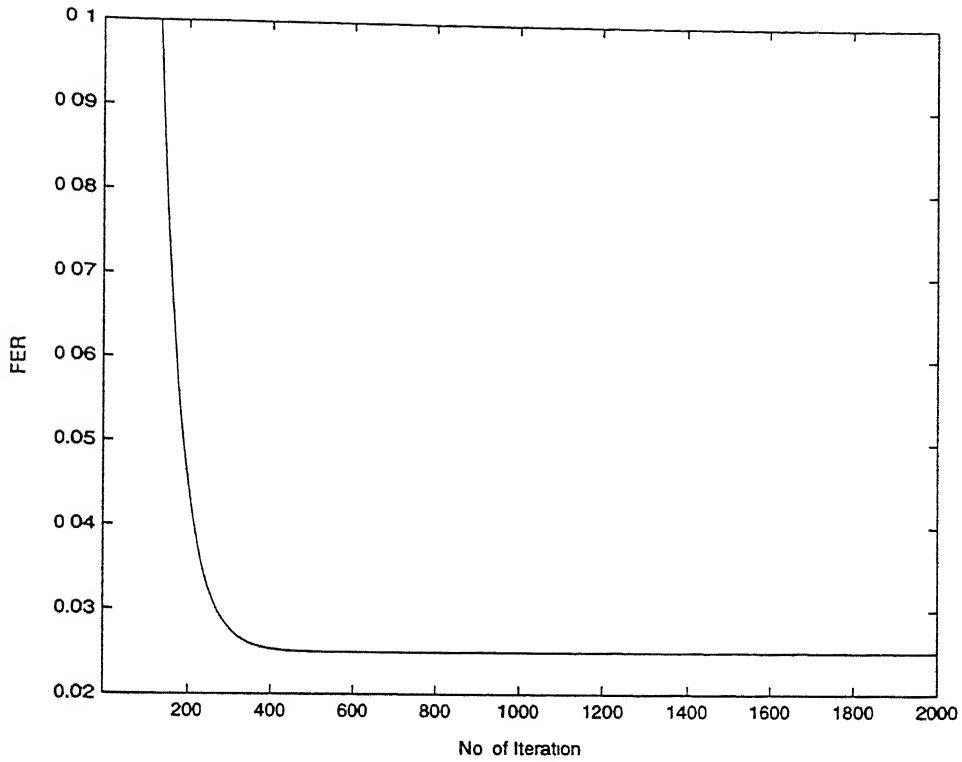


Figure 5.2: Frame error rates versus No. of Iterations for Rate 1/4 GF for Eb/No=1.5 dB in the AWGN channel.

in terms of FER by increasing iterations beyond 500. We observe the same situation for other E_b/N_o. Hence in all of our simulation for $GF(4)$ codes in AWGN channel we have used a maximum of 500 iterations.

In our simulations we did not observe any undetected error which is in agreement with [8], i.e. all decoding errors resulted from non-convergence (we declare non-convergence if the number of iterations reaches the upper limit that in our simulations was set at 500. in contrast with [7] where it is 2000 for binary codes).

Fig. 5.3 shows the performance of $GF(4)$ code for different E_b/N_o. The figure also

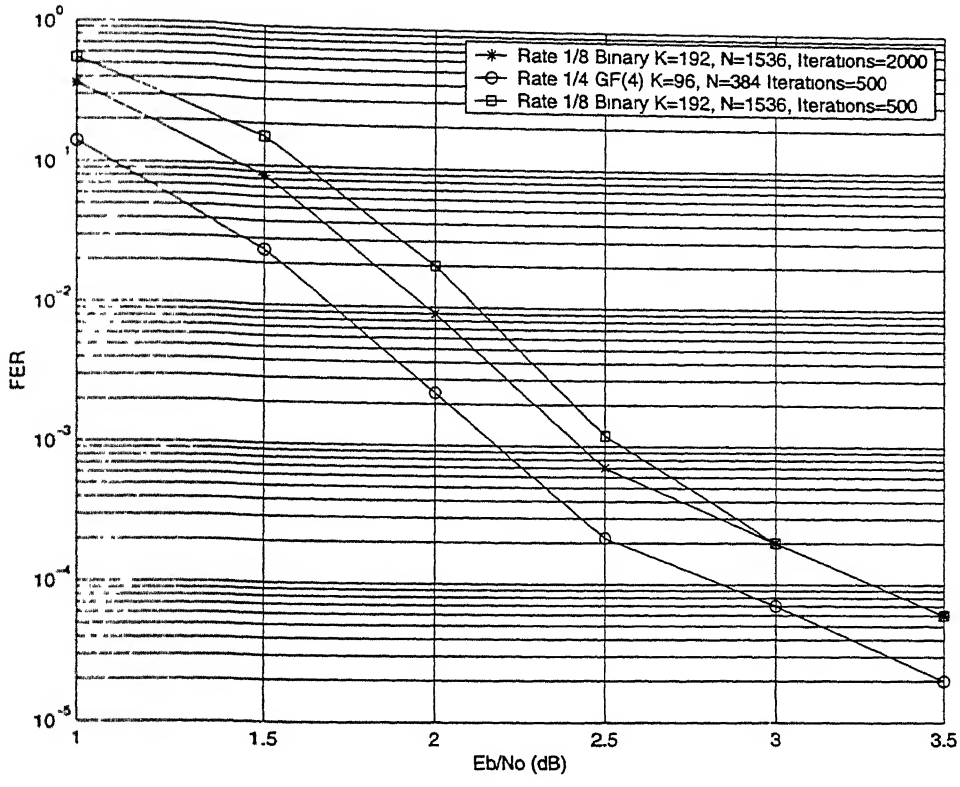


Figure 5.3: Frame error rates for LDPC Codes in the AWGN channel.

shows the performance of binary code with 500 and 2000 iterations. It is clear that $GF(4)$ code performs better then the binary code. It is also clear that for binary case better performance can be achieved by increasing the maximum number of iterations to 2000 as shown in Fig. 5.3, therefore in all the subsequent simulations we have used 2000 iterations for binary codes.

We have already found the per iteration per bit complexity of LDPC codes in chapter 3. However, from the simulations it appears that this is not an adequate indicator of the actual complexity because the number of iterations required per frame is itself a random number. As such more useful benchmark for the complexity specially

from the viewpoint of practical applications, can be found by investigating the average number of iterations required per frame.

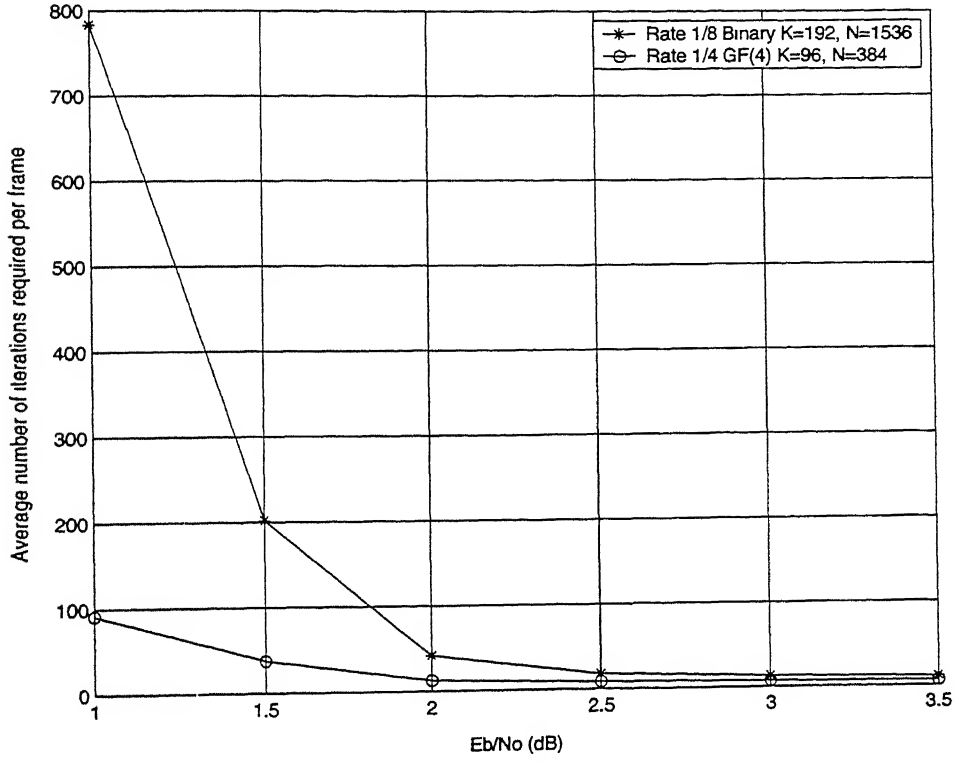


Figure 5.4: Average number of iterations per frame in AWGN channel

Fig. 5.4 shows the average number of iterations that were required in our simulations for decoding $GF(4)$, and binary LDPC codes. (Average number of iteration also includes the number of iterations for the frames that do not converge i.e. detected errors, which is 500 for $GF(4)$ codes, and 2000 for binary codes) Fig. 5.5 shows the average number of iterations only for successfully decoded frames Now overall complexity of $GF(4)$ codes can be found by multiplying the complexity per iteration per information bit by the number of average iterations. Fig. 5.6 shows the overall com-

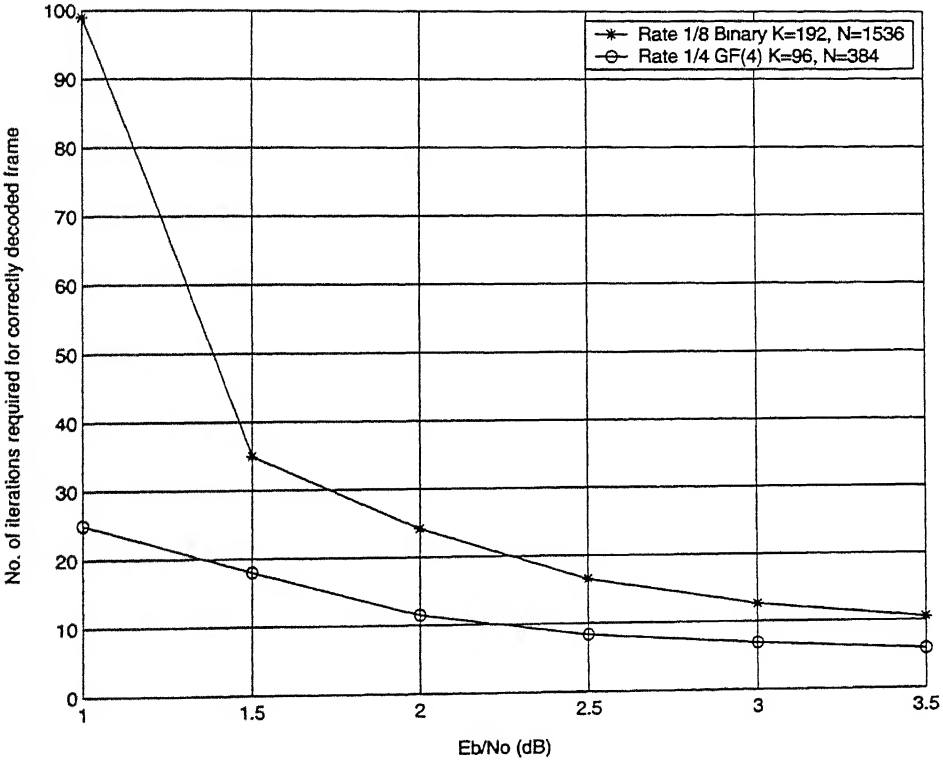


Figure 5.5: Average number of iterations for correctly decoded frame in AWGN channel

plexity i.e. operations required per iteration per information bit $\times$ average number of iterations. To compute overall complexity, the average number of iterations has been taken from Fig. 5.4 Fig. 5.6 also shows the complexity of binary codes for comparison. It is clear from Fig. 5.6 that number of operations required to decode rate $1/4$ $GF(4)$ codes are very less as compared to rate $1/8$ binary codes.

On the contrary if we wish to compare the worst case complexity that a decoder has to reserve, we find that $GF(4)$ codes, outperform binary codes by a huge margin. Thus the complexity for $GF(4)$ codes is $177 \times 500 = 88500$ per bit while that for binary codes is $162 \times 2000 = 324000$ per bit.

5.2 Rayleigh Fading Channel

We also simulated the performance in flat Rayleigh-fading channel. We assumed that fading amplitudes at different time instants are independent Rayleigh random variables with a unit second moment.

We have used the channel model as described in section 3.4.

As in section 3.4, for fading channel we have considered two different cases, one when α i.e. fading amplitude is known to receiver, and other when α is unknown to the receiver

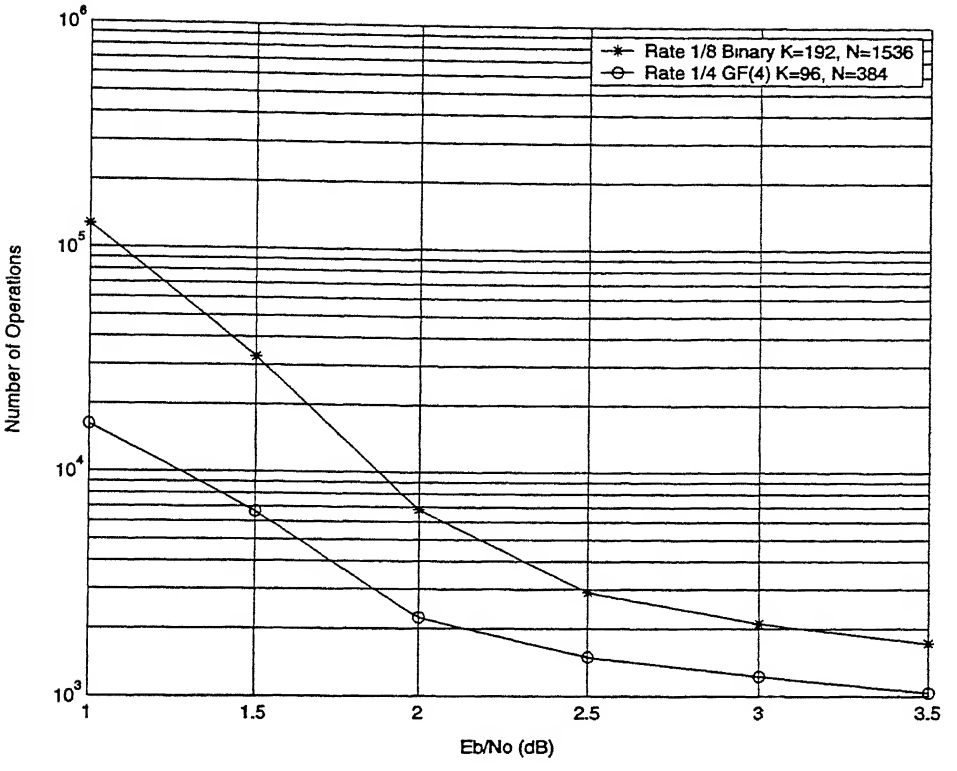


Figure 5.6: Average number of operations required per information bit for AWGN channel.

5.2.1 CSI Available

In this case we assume that the channel state information (CSI) is available to the decoder, i.e. the decoder knows the fading amplitudes α for each transmitted symbol.

Channel likelihood for this case can be found from section 3.4.1.

Fig. 5.7 shows the effect of number of iterations on FER for E_b/N_o=3 dB. It is clear from the figure that we can not gain any significant advantage in terms of FER by increasing iterations beyond 1000. We observe the same situation for other E_b/N_o Therefore we set maximum number of iterations to be 1000 for $GF(4)$ codes in case of

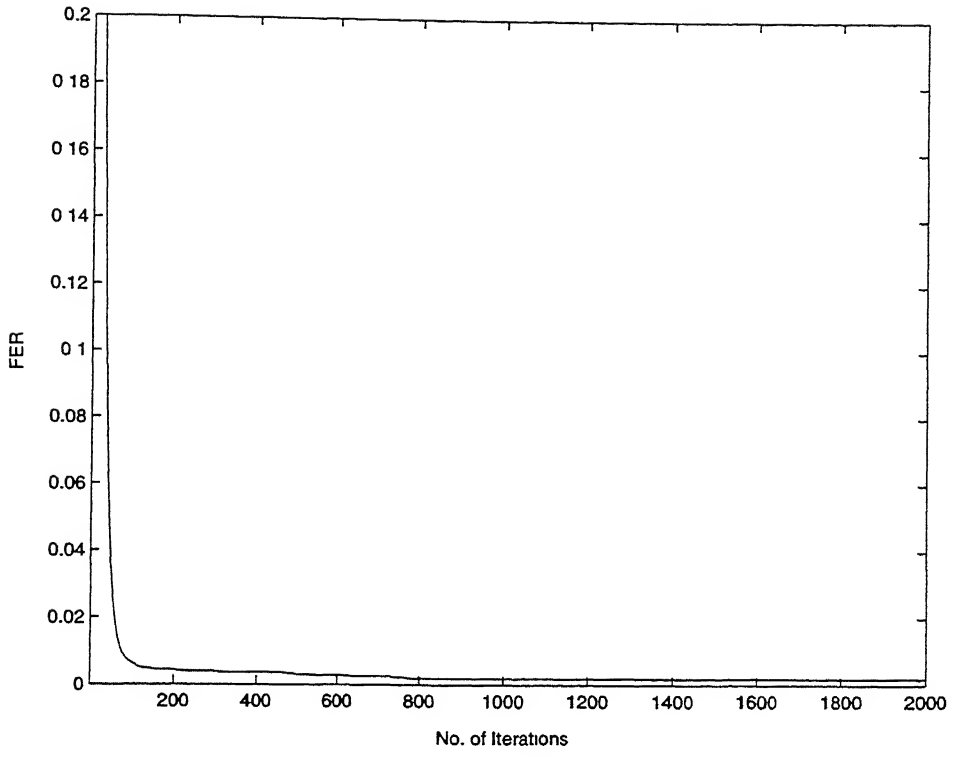


Figure 5.7: Frame error rates versus No. of Iterations for Rate 1/4 GF(4) LDPC codes for Eb/No=3 dB in Rayleigh fading channel. The CSI is available to the decoder.

fading channels.

Fig. 5.8 shows the performance of the $GF(4)$ codes on Rayleigh Fading Channel for different E_b/N_o. Figure also shows the performance of binary rate 1/8 codes. We see that binary codes give a slightly better performance. However this channel model is quite far from reality.

Let us observe the complexity involved in the decoding of LDPC codes. Fig. 5.9 shows the average number of iterations for decoding for $GF(4)$ LDPC codes, and for binary codes. Fig. 5.10 shows the average number of iterations only for successfully

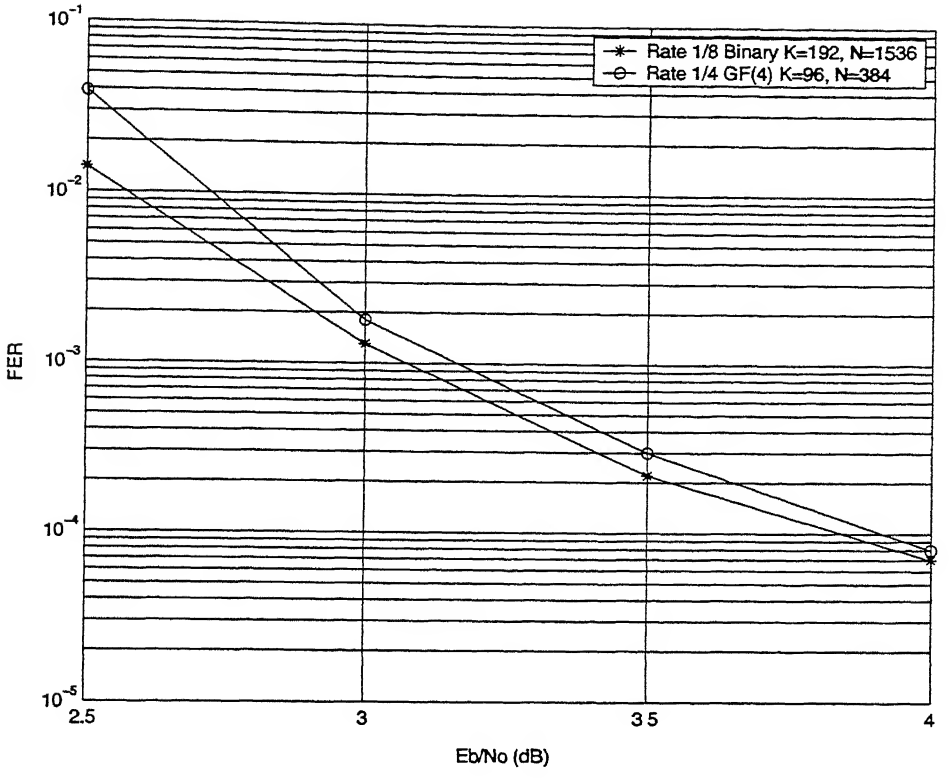


Figure 5.8: Frame error rates for LDPC Codes in the flat Rayleigh fading channel. The channel state information is made available to the decoder.

decoded frame. Fig. 5.11 shows the overall complexity i.e. average number of operations required per information bit. Fig. 5.11 also shows the complexity of binary codes for comparison. It is clear from Fig. 5.11 that number of operations required to decode rate 1/4 $GF(4)$ codes are very less as compared to rate 1/8 binary codes.

5.2.2 CSI Not Available

This is a more realistic situation, since generally receiver does not have knowledge of fading amplitudes. Instead we assume that the receiver knows only the second moment of fading (i.e. $E[\alpha^2]$).

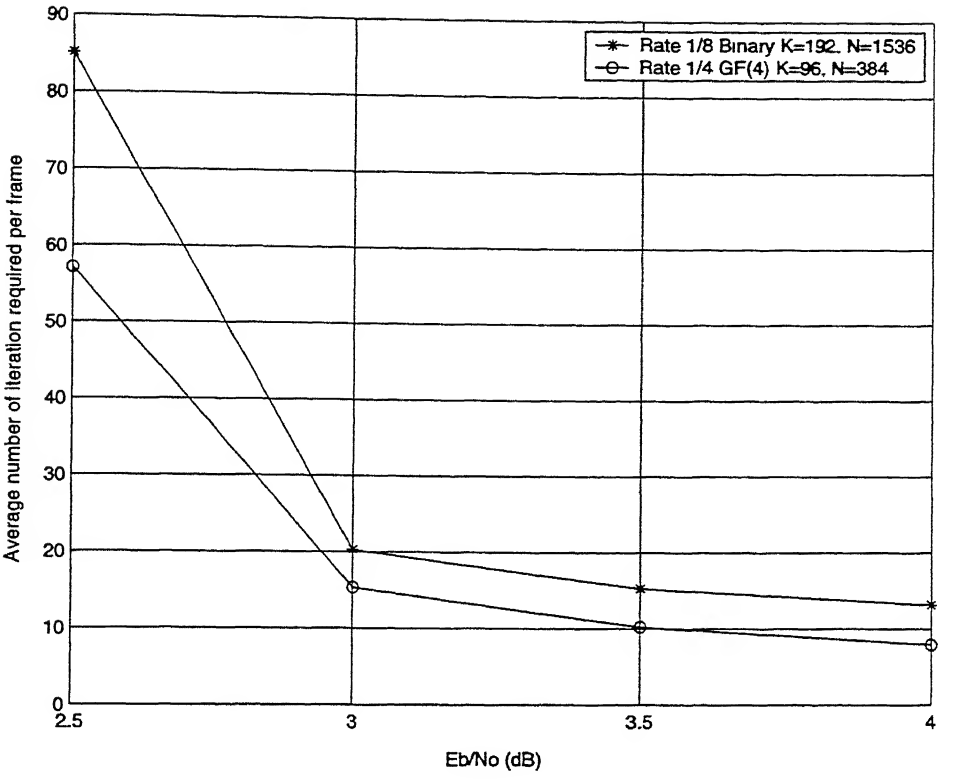


Figure 5.9: Average number of iterations per frame in Rayleigh fading channel. The CSI is available to the decoder

For simulation we have used the channel likelihoods as derived in section 3.4.2.

Fig. 5.12 shows the performance of the $GF(4)$ codes on Rayleigh Fading Channel for different E_b/N_o. The figure also shows the performance of binary codes. We can see that $GF(4)$ codes perform better at higher SNR. However at lower SNR the binary codes perform slightly better.

Let us observe the complexity involved in the decoding of LDPC codes. Fig. 5.13 shows the average number of iterations in decoding for $GF(4)$ and binary codes. Fig. 5.14 shows the average number of iterations only for successfully decoded frame. Fig

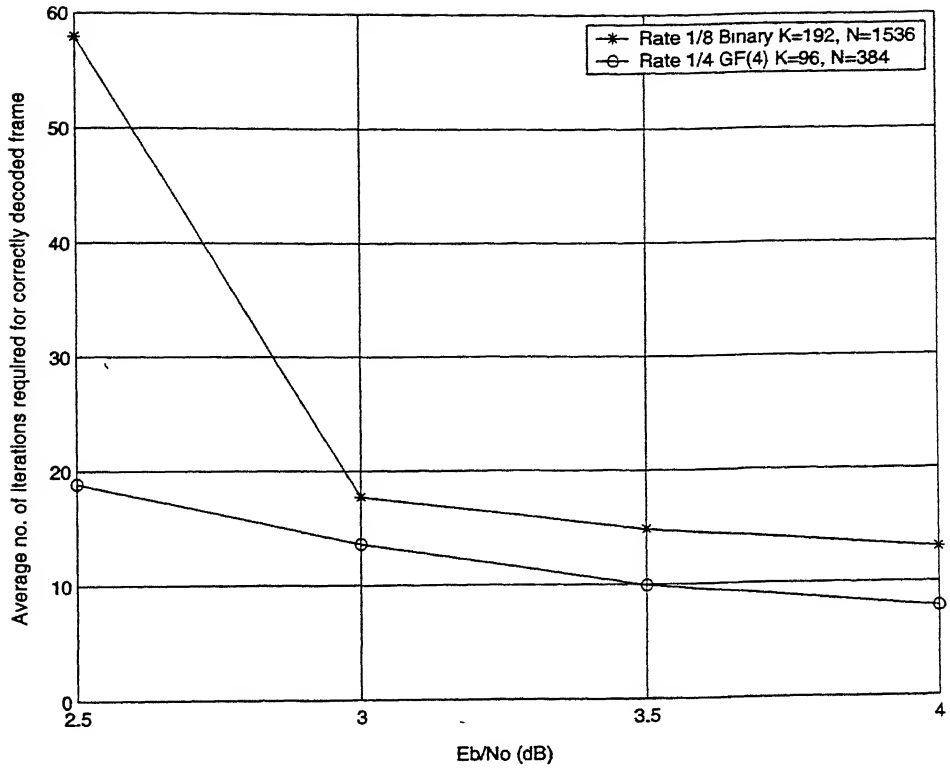


Figure 5.10: Average number of iterations required for correctly decoded frame in Rayleigh fading channel. The CSI is available to the decoder.

5.15 shows the overall complexity in terms of average number of operations per information bit for these codes. Fig. 5.15 also shows the complexity of binary codes for comparison. It is clear from Fig. 5.15 that number of operations required to decode $GF(4)$ codes are very less as compared to binary codes.

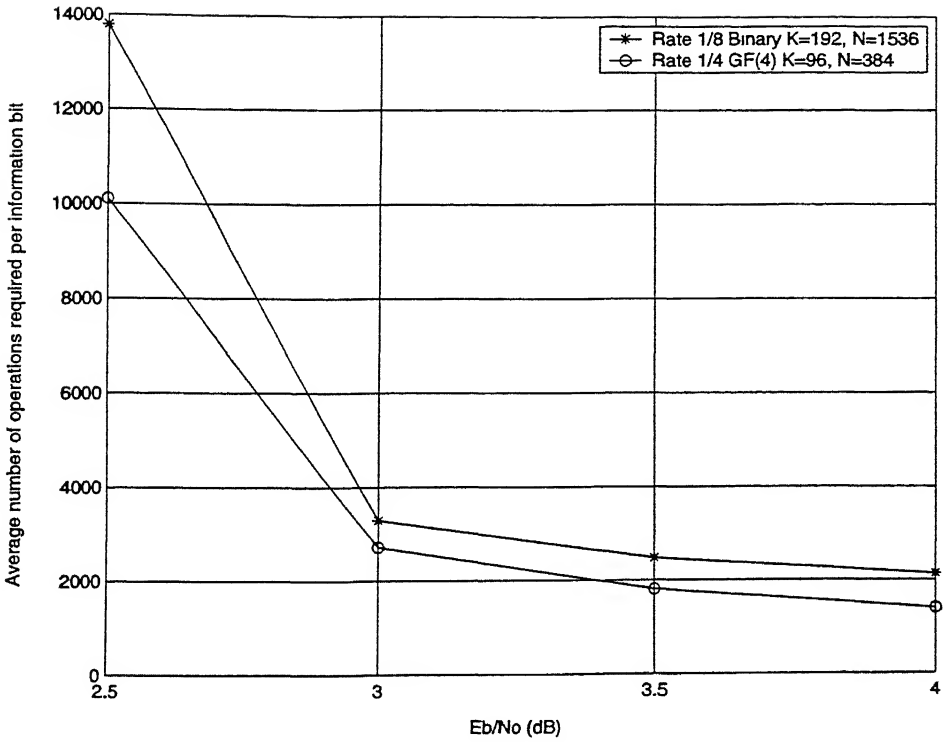


Figure 5.11: Average number of operations required per information bit for Rayleigh fading channel. The CSI is available to the decoder.

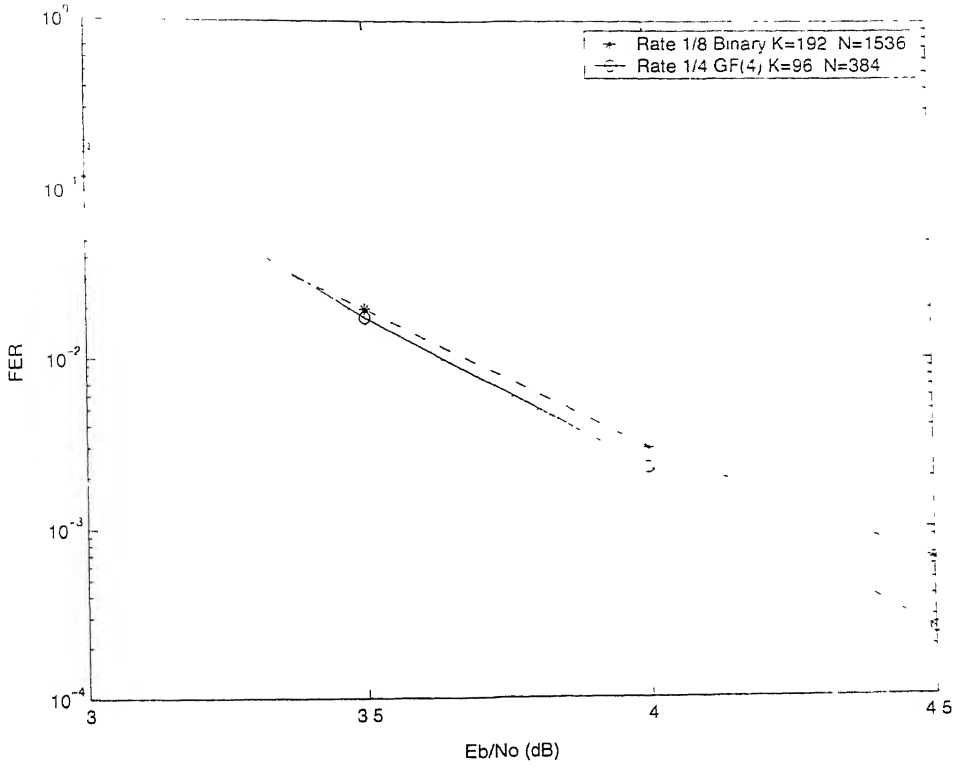


Figure 5.12: Frame error rates for LDPC Codes in the flat Rayleigh fading channel with no CSI

5.3 Capacity of a CDMA Cell

It is known [11] that system capacity of the reverse CDMA link assuming single cell and perfect receiver synchronization, can be approximated as

$$n \approx \frac{\eta}{E_b/I_o} \tag{5.2}$$

where n is the number of users, η is the processing gain and E_b/I_o is the signal-to-interference ratio

We assume perfect synchronization and coherent detection in both forward and reverse links. In order to make a fair comparison with the code in [7] we have used

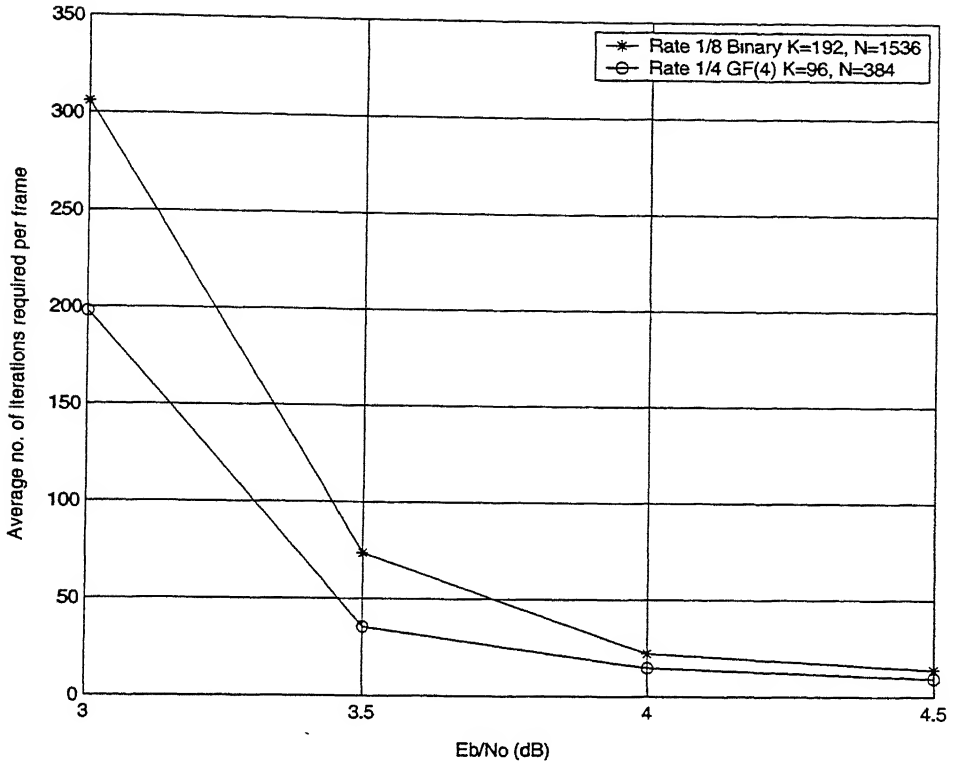


Figure 5.13: Average number of iterations per frame in Rayleigh fading channel with no CSI

$\eta = 64$. So total spreading is 64 Since rate 1/4 codes spread by a factor of 4, rest of the spreading by a factor of 16 can be achieved by multiplying codewords by PN sequence.

The system capacity can be found using (5.2) and Fig. 5.2. Single Cell capacity of a CDMA system with rate 1/4 $GF(4)$ LDPC codes is shown in Fig. 5.16. For comparison the figure also shows the capacity for binary LDPC codes, and low rate orthogonal convolutional codes [12] It is clear from the figure that $GF(4)$ rate 1/4 LDPC codes perform better than binary LDPC codes, and low rate orthogonal convolutional codes

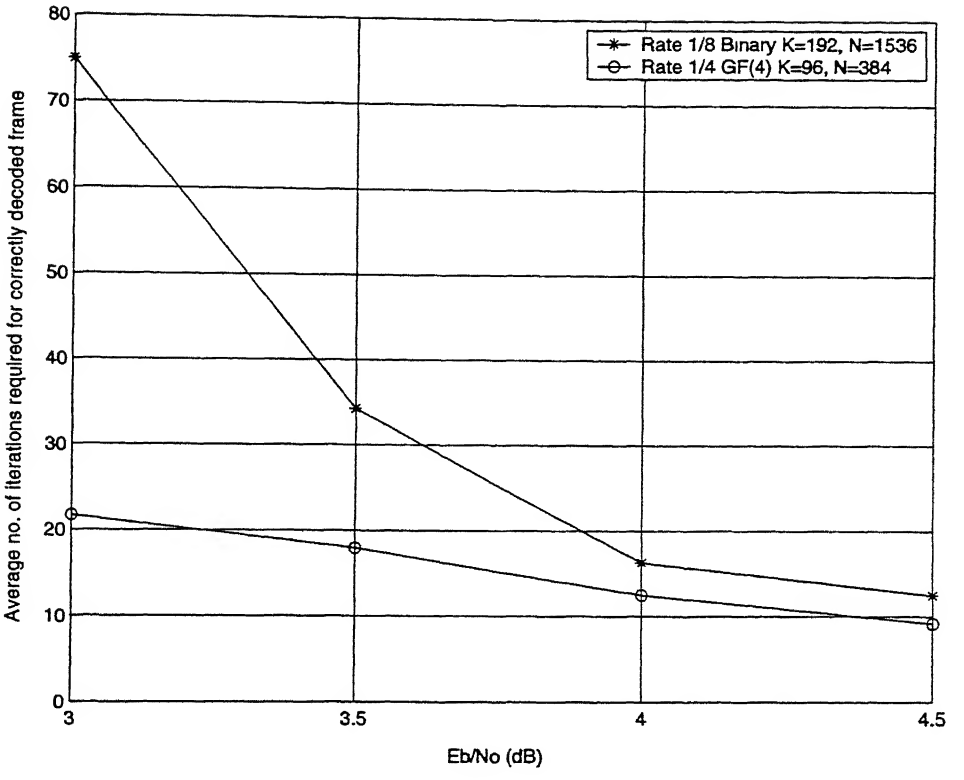


Figure 5.14: Average number of iterations required for correctly decoded frame in Rayleigh fading channel with no CSI

[12], in a multi-user environment.

5.4 Conclusions

In this work we have derived the expression for complexity involved in decoding of non-binary irregular LDPC codes using sum-product algorithm. We have also discussed the hardware implementation of the decoder for these codes.

For the purpose of simulations, the frame length has been chosen to be compatible with IS-95 cellular CDMA system. We have shown that rate $1/4$ $GF(4)$ irregular LDPC

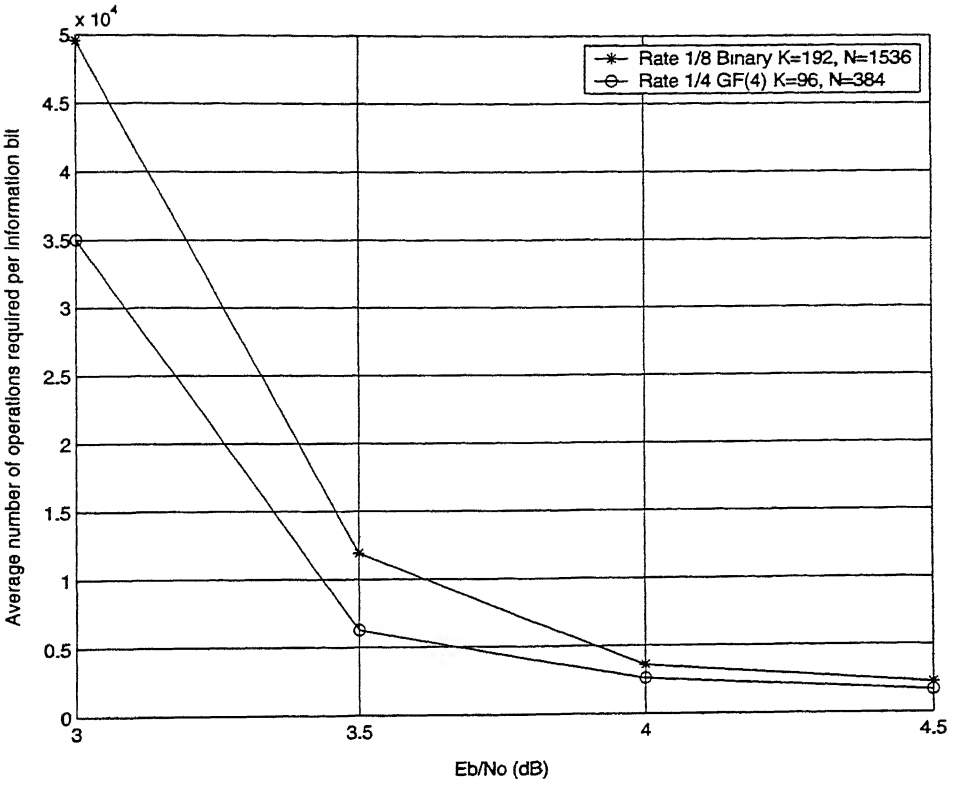


Figure 5.15: Average number of operations required per information bit for Rayleigh fading channel with no CSI

codes can outperform rate 1/8 binary LDPC codes both in terms of FER performance and decoding complexity in an AWGN channel. Thus if both are provided the same complexity, $GF(4)$ codes will give a much better FER, while if both have to give same FER $GF(4)$ will provide at much lower complexity. But more importantly if FER is the only criteria, binary codes may not be able to achieve the FER achieved by $GF(4)$ codes

However the results for FER are not as good in the case of Rayleigh channel

We have also evaluated the enhanced capacity that can be provided by $GF(4)$ codes

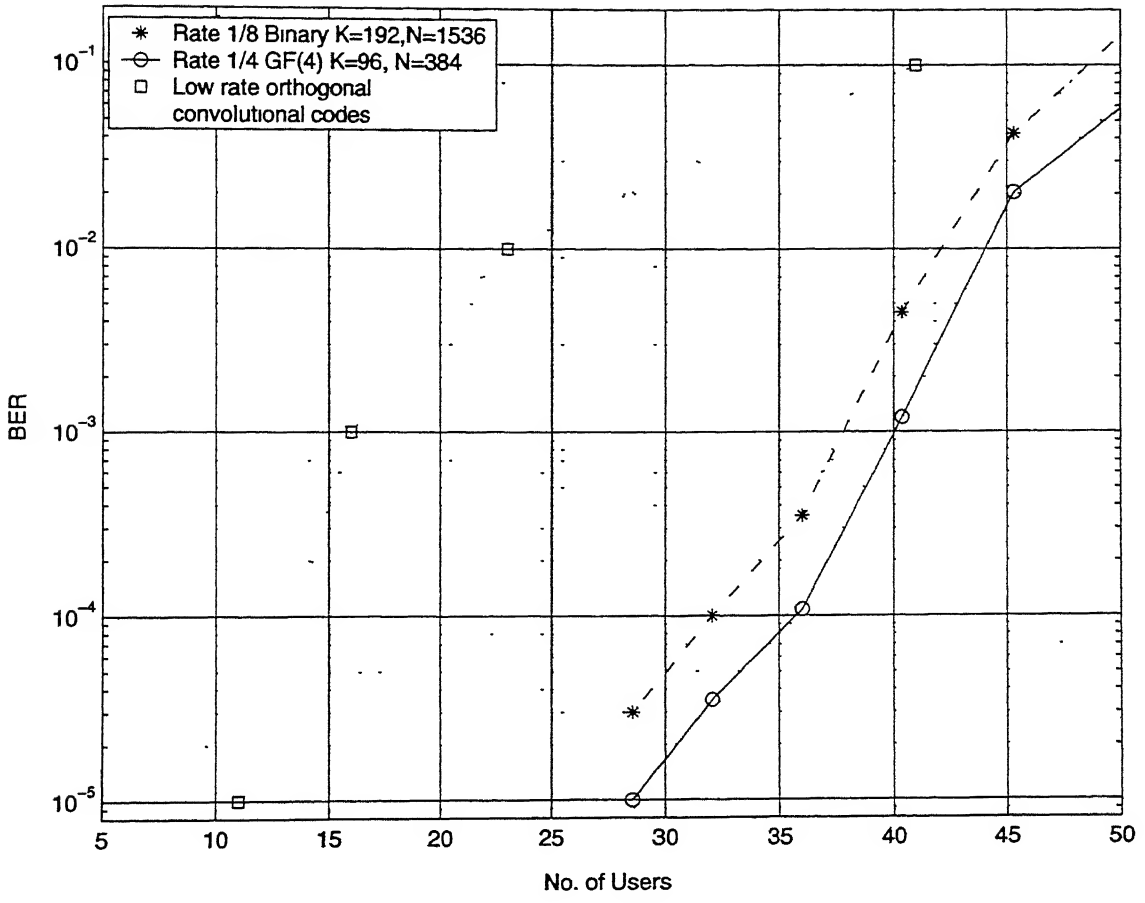


Figure 5.16: Data BER versus number of users in a cell for $GF(4)$ rate 1/4 code, binary rate 1/8 code[7], low-rate orthogonal convolutional code [12]

in a single cell CDMA environment.

5.5 Scope for Future Work

This work need to be extended to non-binary LDPC codes. Specially we need to investigate the performance of codes over $GF(8)$ and other higher fields.

We have used binary modulation schemes. It may be worthwhile to investigate the

performance using higher order modulations.

LDPC codes are known to be sensitive to frame lengths. Hence we need to find out the range of frame lengths over which $GF(4)$ codes outperform binary codes.

References

[1] R.G. Gallager, *Low-Density Parity-Check Codes.* Cambridge, MA: MIT Press, 1963.

[2] A.J. Viterbi, "Very low rate convolutional codes for maximum theoretical performance of spread-spectrum multiple-access channels,"*IEEE J. Select. Areas Communications*, vol. 8, pp. 641-649, May 1990.

[3] R.M. Tanner, "A recursive approach to low complexity codes,"*IEEE Transactions on Information Theory*, vol. IT-27, pp. 533-547, Sept. 1981.

[4] M.G. Luby, M. Mitzenmacher, A. Shokorollahi, and D.A. Spielman, "Improved low density parity check codes using irregular graphs,"*IEEE Transactions on Information Theory*, vol47, No. 2, pp. 585-598, Feb. 2001.

[5] J.S.Lee , L. E. Miller, *CDMA Systems Engineering Handbook.* Reading, Artech House, Incorporated. 1998

[6] V Sorokine, F. R. Kschischang and S. Pasupathy, "Gallagar codes for CDMA applications-Part I. Generalizations, Constructions, and Performance

Bounds,"*IEEE Transactions on Communications*, vol. 48, pp. 1660-1667, Oct. 2000.

[7] V. Sorokine, F. R. Kschischang and S. Pasupathy, "Gallagar codes for CDMA applications-Part II: Implementations, Complexity, and System Capacity,"*IEEE Transactions on Communications*, vol. 48, pp. 1818-1828, Nov. 2000.

[8] M.C. Davey and David MacKay, "Low-density parity check codes over GF(q),"*IEEE Communication Letters* , vol. 2, No. 6, pp. 165-167, June 1998.

[9] D.J.C. MacKay, "Good error-correcting codes based on very sparse matrices,"*IEEE Transactions on Information Theory*, vol. 45, pp. 399-431, March 1999.

[10] M. C. Davey, *Error-correction using Low-Density Parity-Check Codes*, Ph.D. dissertation, Inference Group, Cavendish Laboratory, University of Cambridge, December 1999.

[11] A.J. Viterbi, *CDMA Principles of Spread Spectrum Communication.* Reading, MA: Addison-Wesley, 1995.

[12] R.F. Ormondroyd and J.J. Maxey, "Performance of low-rate orthogonal convolutional codes in DS-CDMA environment,"*IEEE Transactions on Vehicular Technology*, vol. 46, pp. 320-328, May 1997.